SMART APPLIANCES SCHEDULING FOR SMART GRID

INTRODUCTION

With the advent of Smart Appliances and smart grids, it is safe to say, that the new paradigm will be to schedule our Appliances in advance with reference to the Smart Grid's day look ahead prices. This will help, in not only reducing the total cost of the electricity bill, but will also help the Electricity supplier to effectively plan and reduce his additional consumption from the main grid. The user (House) can enjoy the benefits of the already implemented schedule, as he will be charged the same price as per his committed schedule, till he decides to change the schedule again. This implementation will allow multiple Houses connected to common Utility server and respective Home Servers to schedule their appliances according to their own requirement.

1. ARCHITECTURE

The user has the schedule accessible on his Mobile via an application, which will enable him to remotely login and either check his current schedule or request a new schedule. The implementation or basic architecture (Fig 1) is as follows:

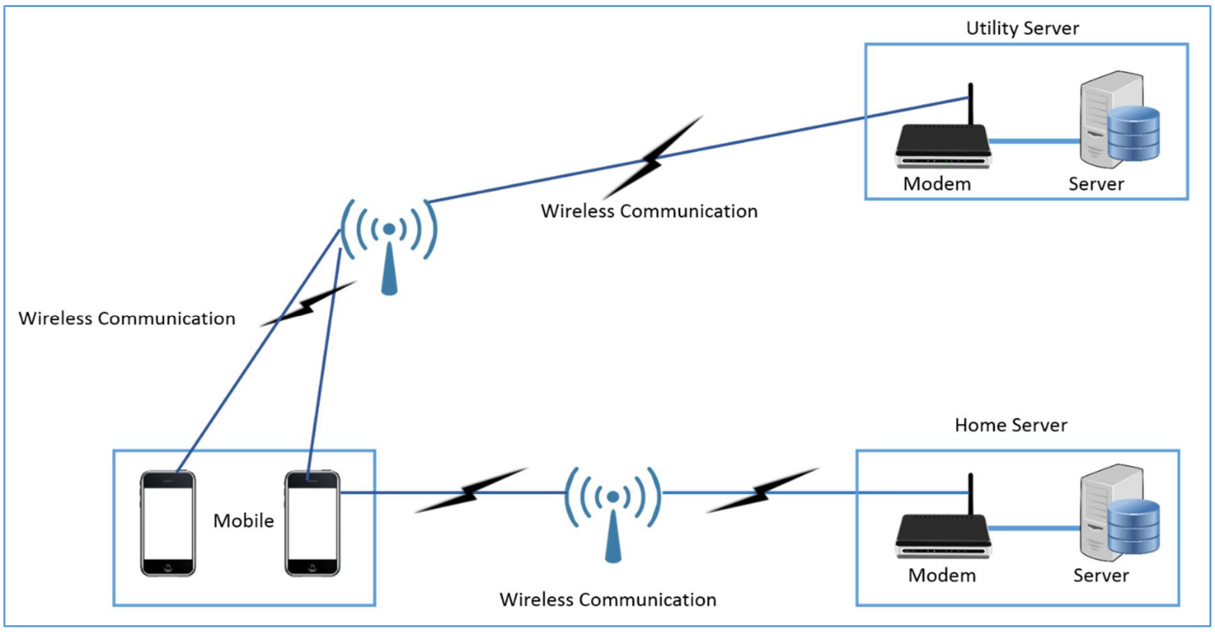
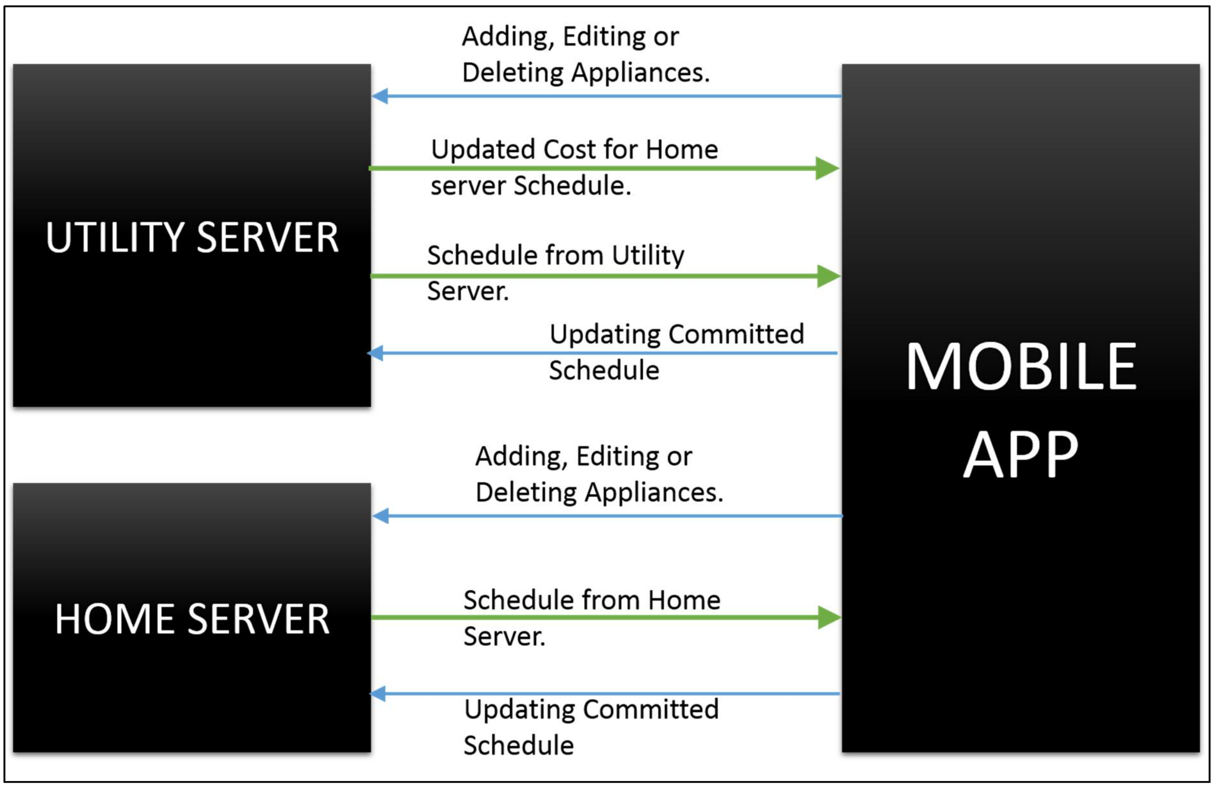


Fig 1: System Architecture

The user via his mobile app has the option to either view the current schedule or request for a new schedule. He has the option to either query the new Schedule from the Home server or from Utility server. Once he receives a schedule from Home server, he can verify the cost of the schedule by verifying the values from the Utility server. The Utility server will send back the schedule with updated costs if any.

The Flow of the system is as given below:


UTILITY SERVER
HOME SERVER
MOBILE APP
Adding, Editing or Deleting Appliances.
Updated Cost for Home server Schedule.
Schedule from Utility Server.
Updating Committed Schedule
Adding, Editing or Deleting Appliances.
Schedule from Home Server.
Updating Committed Schedule

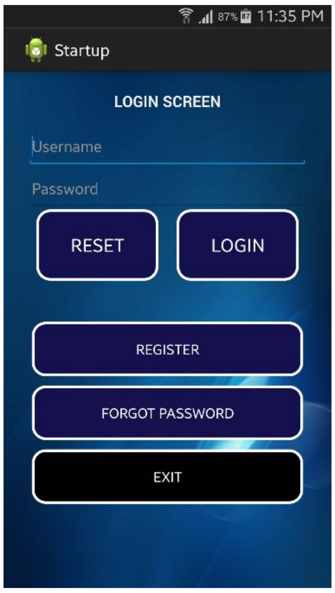
The project is divided into 3 sections:

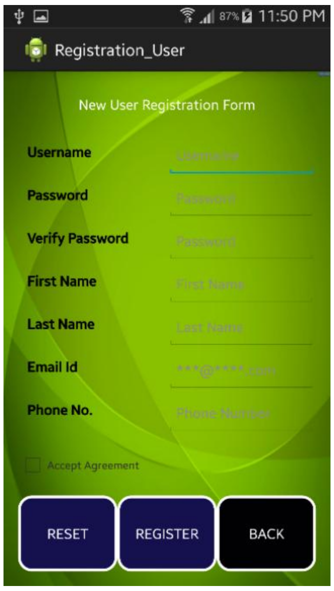
- Mobile App
- Home Server
- Utility Server

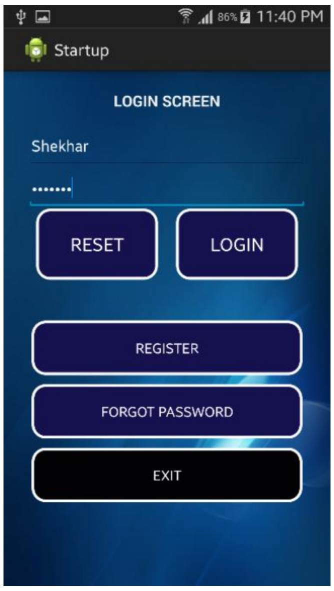
2. MOBILE APP

2.1 Design

1. The User will login to his app using his credentials, i.e. Userid and Password, if it's the first time, then the user can click on Register and create a new userid and password. Every House can only have one unique userid. The process is as follows:



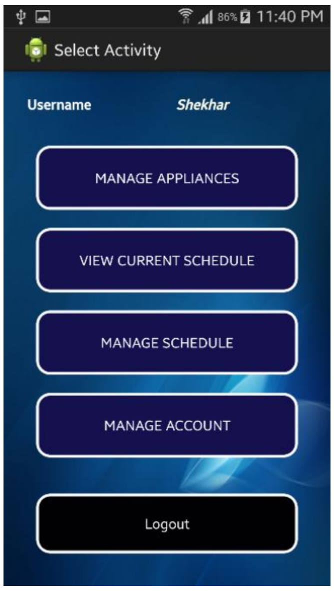




Shekhar

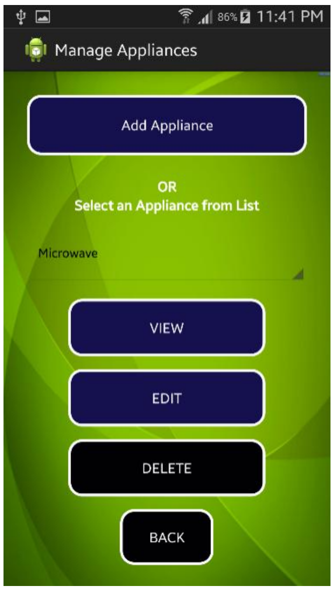
•••••••

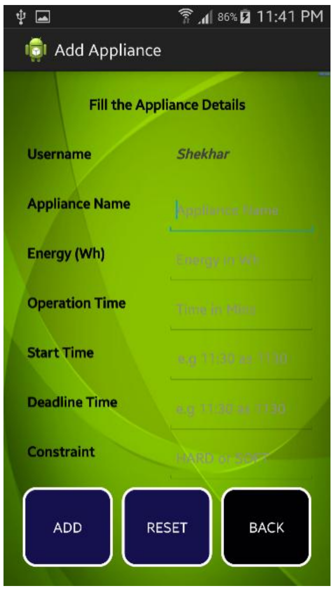
2. Once the user is able to successfully login, he can select from various activities:



Username *Shekhar*

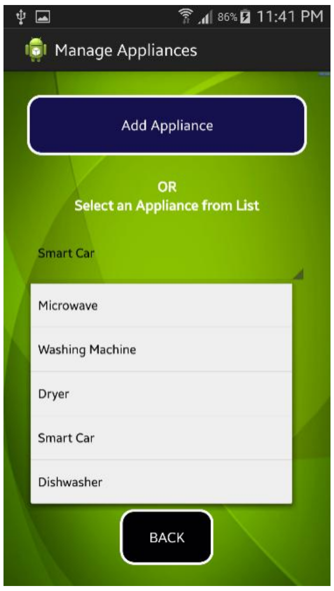
3. The user should first populate his Appliances list. He can add new appliances if required, or view an appliance by selecting the appliance from the drop down list, or he can edit the previously entered details or he can simply delete the appliance.

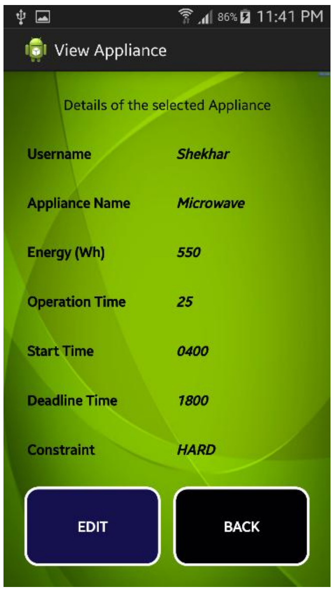



Add Appliance

Fill the Appliance Details

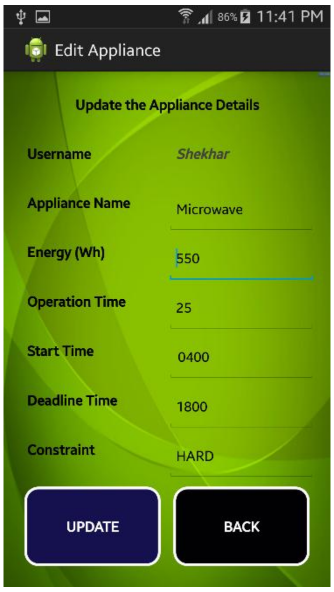
Username	*Shekhar*
Appliance Name	Appliance Name
Energy (Wh)	Energy in Wh
Operation Time	Time in Mins
Start Time	e.g 11:30 as 1130
Deadline Time	e.g 11:30 as 1130
Constraint	HARD or SOFT




View Appliance

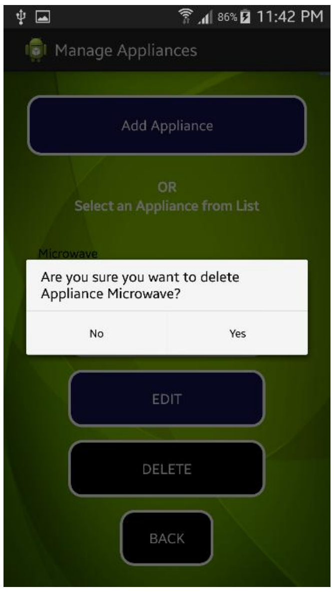
Details of the selected Appliance

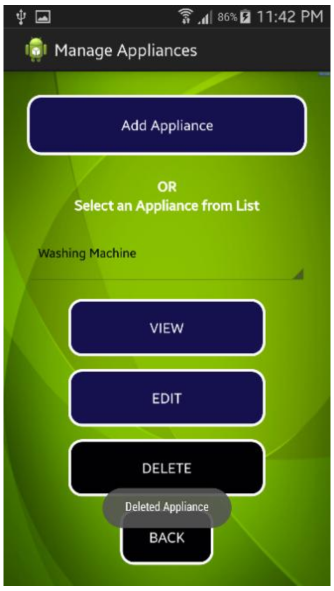
Username	*Shekhar*
Appliance Name	*Microwave*
Energy (Wh)	*550*
Operation Time	*25*
Start Time	*0400*
Deadline Time	*1800*
Constraint	*HARD*


Edit Appliance

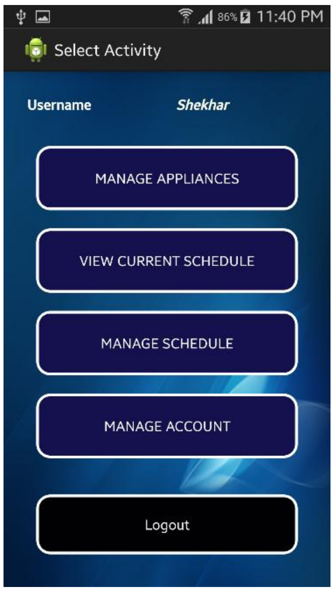
Update the Appliance Details

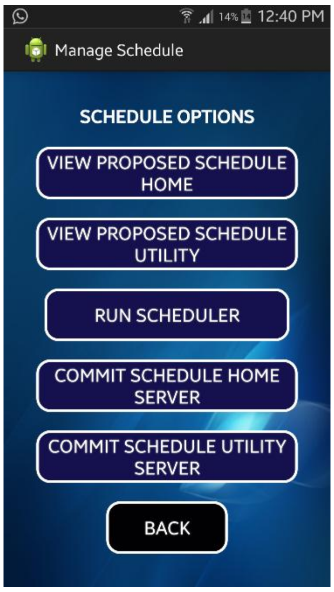
Username	*Shekhar*
Appliance Name	Microwave
Energy (Wh)	550
Operation Time	25
Start Time	0400
Deadline Time	1800
Constraint	HARD


Are you sure you want to delete Appliance Microwave?

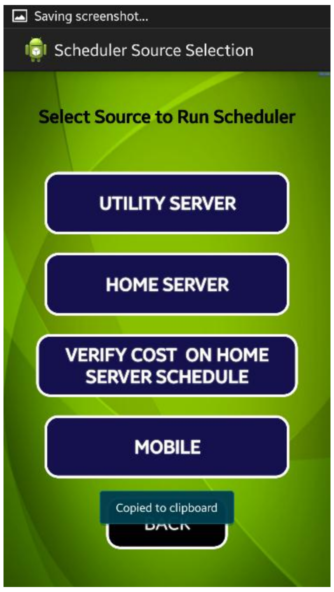

Deleted Appliance

4. Once the appliance Database is populated, now the user should run his Scheduler for the first time. The user has options to run the scheduler on either the Home server or the Utility server.
5. If the user selects Home server, he can view the proposed schedule by going back and selecting 'VIEW PROPOSED SCHEDULE HOME'.
6. If the Scheduler is run from Home server, the user can select 'VERIFY COST ON HOME SERVER SCHEDULE' to send the data to Utility server, which will send the updated Cost for the Schedule.
7. If the Scheduler is run from Utility, the schedule is received and the user can view it.

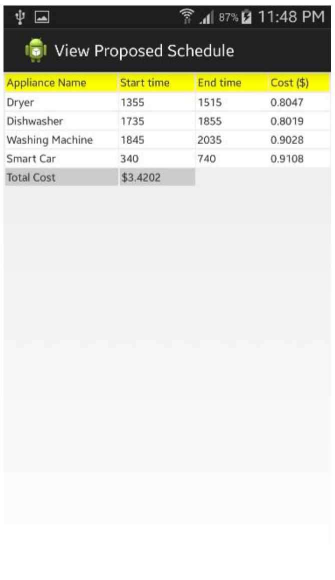




14% 12:40 PM
Manage Schedule
SCHEDULE OPTIONS
VIEW PROPOSED SCHEDULE HOME
VIEW PROPOSED SCHEDULE UTILITY
RUN SCHEDULER
COMMIT SCHEDULE HOME SERVER
COMMIT SCHEDULE UTILITY SERVER
BACK

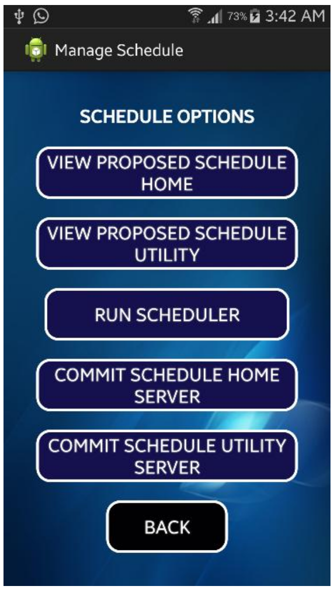


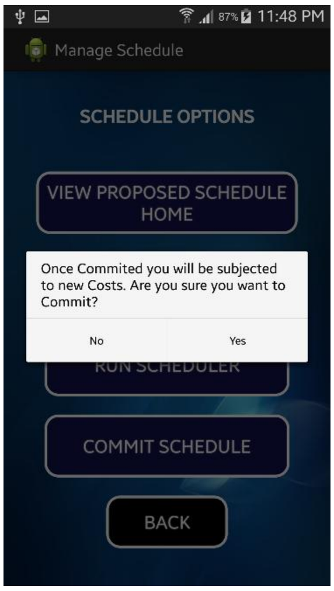
Saving screenshot...
Scheduler Source Selection
Select Source to Run Scheduler
UTILITY SERVER
HOME SERVER
VERIFY COST ON HOME SERVER SCHEDULE
MOBILE
Copied to clipboard
BACK



Appliance Name	Start time	End time	Cost ($)
Dryer	1355	1515	0.8047
Dishwasher	1735	1855	0.8019
Washing Machine	1845	2035	0.9028
Smart Car	340	740	0.9108
Total Cost	$3.4202		

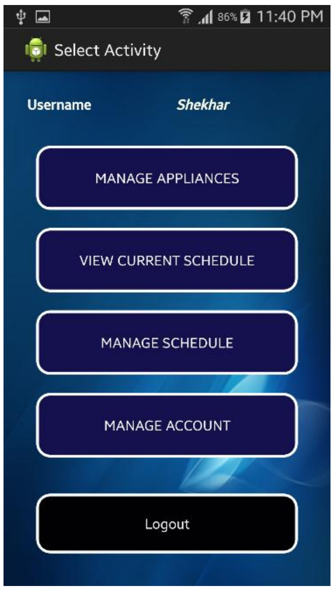
8. Once the schedule is received, the user has option to either commit it or discard it.

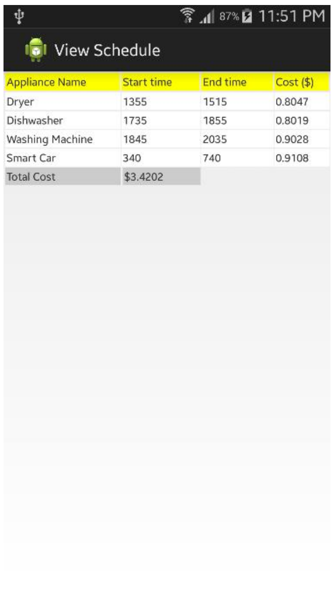

3:42 AM
Manage Schedule
SCHEDULE OPTIONS
VIEW PROPOSED SCHEDULE HOME
VIEW PROPOSED SCHEDULE UTILITY
RUN SCHEDULER
COMMIT SCHEDULE HOME SERVER
COMMIT SCHEDULE UTILITY SERVER
BACK



Once Commited you will be subjected to new Costs. Are you sure you want to Commit?

9. The final Schedule can be viewed from the original menu option.



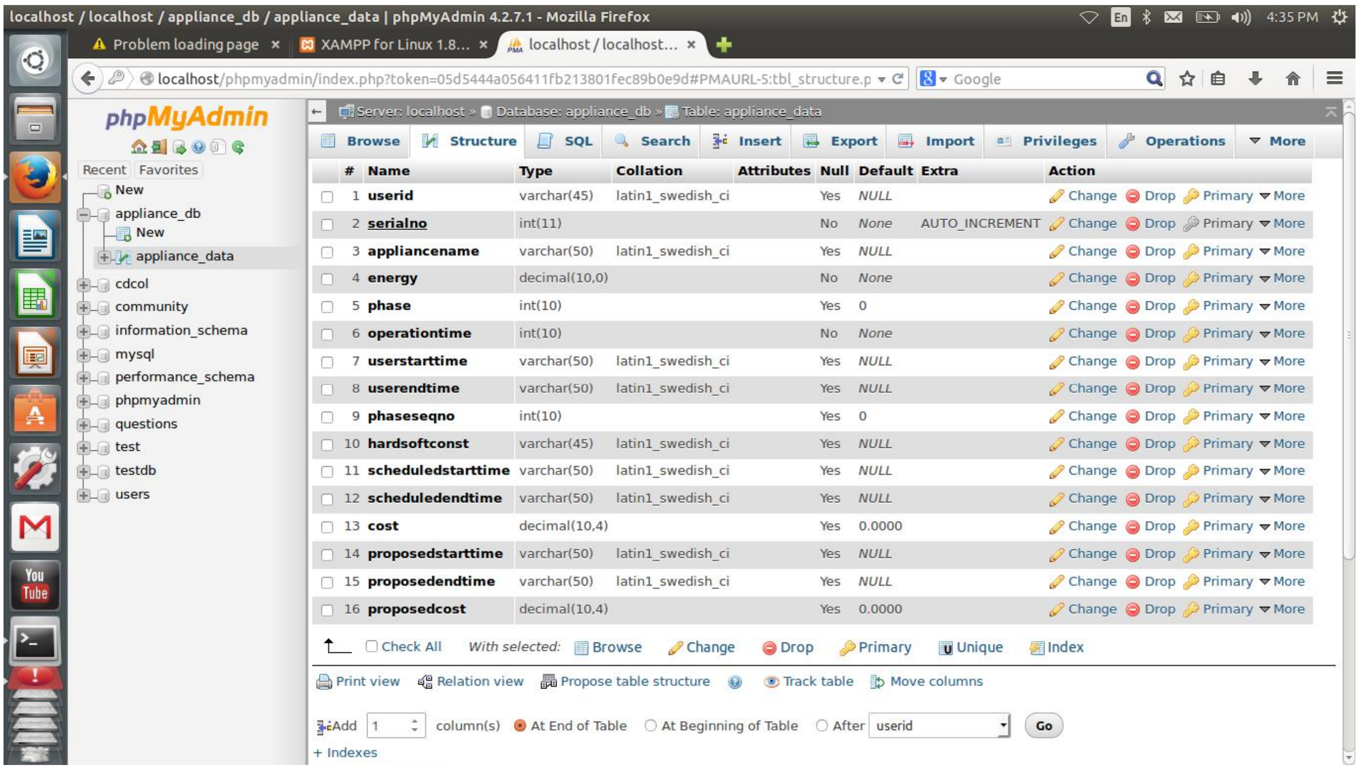


Appliance Name	Start time	End time	Cost ($)
Dryer	1355	1515	0.8047
Dishwasher	1735	1855	0.8019
Washing Machine	1845	2035	0.9028
Smart Car	340	740	0.9108
Total Cost	$3.4202		

2.2 Design Considerations and Assumptions

The major design considerations and assumptions are as follows:

1. The Home server and Utility server, both have independent databases. The structure of the database is :



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	**userid**	varchar(45)	latin1_swedish_ci		Yes	*NULL*		Change Drop Primary More
2	**serialno**	int(11)			No	*None*	AUTO_INCREMENT	Change Drop Primary More
3	**appliancename**	varchar(50)	latin1_swedish_ci		Yes	*NULL*		Change Drop Primary More
4	**energy**	decimal(10,0)			No	*None*		Change Drop Primary More
5	**phase**	int(10)			Yes	0		Change Drop Primary More
6	**operationtime**	int(10)			No	*None*		Change Drop Primary More
7	**userstarttime**	varchar(50)	latin1_swedish_ci		Yes	*NULL*		Change Drop Primary More
8	**userendtime**	varchar(50)	latin1_swedish_ci		Yes	*NULL*		Change Drop Primary More
9	**phaseseqno**	int(10)			Yes	0		Change Drop Primary More
10	**hardsoftconst**	varchar(45)	latin1_swedish_ci		Yes	*NULL*		Change Drop Primary More
11	**scheduledstarttime**	varchar(50)	latin1_swedish_ci		Yes	*NULL*		Change Drop Primary More
12	**scheduledendtime**	varchar(50)	latin1_swedish_ci		Yes	*NULL*		Change Drop Primary More
13	**cost**	decimal(10,4)			Yes	0.0000		Change Drop Primary More
14	**proposedstarttime**	varchar(50)	latin1_swedish_ci		Yes	*NULL*		Change Drop Primary More
15	**proposedendtime**	varchar(50)	latin1_swedish_ci		Yes	*NULL*		Change Drop Primary More
16	**proposedcost**	decimal(10,4)			Yes	0.0000		Change Drop Primary More

2. Every event triggers a predefined php script.
3. 21 php scripts have been written for implementing the above project.
4. Both the databases are simultaneously updated, and hence there is no discrepancy of the data.
5. Assumption is there is no direct communication between both the databases and the Mobile app is the only source of sending and receiving data.

3. UTILITY SERVER

The utility server can schedule the appliances for all consumers in its server in a globally optimized way as it has a global view of all schedules in its distribution network. Consumer specific schedules are there after sent back to the respective mobile devices for the consumers to commit or override. The utility server will read detailed grid and appliance models of the Consumer homes/ posted by the mobile app, get real-time power demand and run optimizations at the system level for all Consumer schedules and notify the Consumer mobile devices. A distributed optimization scheme (DOS) is created using the simulated annealing algorithm. The main goal of the utility server is not only to optimize the CPS load scheduling for a particular house but also to reduce the peak CPS demand over all houses.

3.1 Constraints

1. Each appliance has a total energy demand that must be satisfied over all its scheduled instances.
2. Each appliance has a time range of execution bounded by a start-time and a deadline. The final schedule achieved by OPT has to meet these constraints.

3. Appliances can be Soft Real-Time (SRT) constrained or Hard Real-Time constrained (HRT). The SRT appliances can be scheduled from any time after the User preference start time. The tardiness of these appliances (the cost difference of actual scheduled end time and user preference end time) is part of the cost function that should be minimized as well. The HRT appliance needs to be scheduled within the user preference start and end time.

3.2 ASSUMPTIONS

1. The Main database file is posted to the utility server from the mobile app and hence the utility server can read those files.
2. The Peak power database and Look Ahead Day Cost files are updated on a daily basis and can be accessed by the utility server, as it requires.
3. There is no RPS scheme for consumers.

3.3 ALGORITHM

1. Set key as User Id and split the main database according to key.

2. Initialize empty lists for Main database, Look Ahead Day cost, Peak power database.

3. Read the text files and store it to the lists

4. Initialize the temperature to 10000

5. Set initial current cost to 0

6. If temperature is less than 1 go to step 22

7. Call function to create a random schedule

8. Get the user preference start and end times

9. Divide the time from start to end into 5 minute time slots and store them to an array

10. Get a random time from the array

11. Call the function to check if peak power has reached during selected time

12. Check the main database for appliances running in the selected time

13. Compute the total energy of all appliances running in that selected time.

14. If total energy has exceeded the peak power during that time slot then remove that time from array and go back to step 9

15. Assign the selected time to appliance as its proposed start time

16. Calculate the total cost incurred due to all the scheduled appliances

17. If the cost of this schedule is less than current cost go to step 19

18. Check acceptance probability. If less than a random value then go to step 19 else go to step 7
19. Set this as best schedule

20. Cool the system temperature and go to step 6

21. Set the best schedule as the final schedule

22. Stop

4. HOME SERVER

4.1 OVERVIEW

Demand Side Management (DSM) optimization framework – this project, the idea is to optimize the load scheduling of all appliances in all the houses to reduce the peak power demand on the commercial power supply (CPS) at any instance, while simultaneously reducing the energy demand on the CPS by shifting loads onto the RPS of the houses (subject to generation and storage capacities of their respective RPS). Hence, we can achieve reduction of total cost of electricity (T$) to the consumer.
To achieve this goal, we use a distributed optimization scheme (DOS), with all houses in the community have their own copies of an optimization algorithm (OPT) running on their respective servers. The goal of this algorithm is to optimize the CPS load scheduling, for a particular house producing a local optimum schedule. Once the OPT algorithm is executed we check to shift load on to RPS for each of the appliances in the house depending on RPS energy generation and RPS availability. On completion of load transfer to RPS we get new CPS schedule with reduced load requirement and thereby reducing cost.

In order, to get a global optimum schedule each house has to send its schedule to the utility server and utility server runs an OPT algorithm to get global optimum schedule through peak reduction and valley filling mechanism.

Once an OPT at a house is executed on its home server, the schedule of that house is sent via the mobile device to the utility server as to get cost as per updated day-ahead-cost and if its exceeding the peak then corresponding spot price it added to total cost. This updated cost sent by the utility server is received on the users mobile, where the user has the option of accepting the cost for the schedule provided by home server via a COMMIT or he can request rescheduling on the utility server which might yield a better overall cost.

4.2 DESIGN

Consider a DOS when OPT is about to be executed for a house. Using scheduling terminology, we refer to the assignment of an appliance to a certain time instance, as the scheduling of the corresponding job. All such job is tied to a particular set of appliances, have a time range of execution bounded by a start-time and a deadline. The total power of the job is known by adding the power demands of the individual appliances. The energy requirement of an appliance is the product of its power and required run time. The total energy requirement of the job is found by adding the individual energy requirements of its mapped appliances at each of 5 minute intervals between start time and end time. Some appliances {Jx} can be considered Soft Real-Time (SRT) constrained, and they do not have to meet strict deadlines while other may be Hard Real-Time(HRT). The tardiness of these appliances {D(Jx)} is measured by their completed delays beyond their deadlines, are part of the cost function which will be minimized.

In this project, we use Simulated Annealing as our OPT algorithm to schedule all the appliances for a house based on user constraints to get CPS load schedule. The OPT algorithm picks a random start time and end time in each annealing cycle for each appliance. The end time would be end of day i.e 2400Hrs minus the operation time for SRT while end time HRT would user set end time. Then the OPT algorithm calculates the cost for each appliance including the tardiness for each start time and compares with cost of previously selected start time and saves the schedule providing the least cost. Re-scheduling of a job from the current instance to a later one is done to take advantage of a lower pre-dispatch price of power in a future hour compared to the real-time price of power in the current hour. Once the OPT algorithm has completed, we check RPS availability for each appliance based on the RPS energy available and also time. If the RPS is available, we shift the load in increments of 10units for each appliance scheduled at same time onto RPS, so that energy from the RPS is distributed among all the appliances uniformly and thereby a cost reduction is obtained.

4.3 Project Data Set

The OPT algorithm (Simulated Annealing) is implemented in java (hadoop) and executed on single node (pseudo distributed) hadoop (v1.2.1) cluster.

The java code containing mapreduce program is designed using Eclipse Luna.
The OPT algorithm is invoked using bash scripts which for check for schedule requests by a user. For purpose of this project we assume that RPS available probability is 80% and energy available is 300Wh between 6am and 6pm.
The home server is running lampp on Ubuntu-linux with a MYSQL server which contains user appliance database which provides access to user's mobile device to add/delete appliance, request scheduling and display current schedule. The OPT algorithm receives this data via CSV file as input and additionally it has fetch day-ahead-cost from a text file for cost computation.

4.3 PROBLEMS

Hadoop multinode implementation could not be implemented as the LxC container failed to recognize the path of day-ahead-cost file during execution, resulting in File Not Found error. Different approaches to pass the day-ahead-cost file path to the program, such as passing through command line argument, reading from container DFS, and reading from hadoop dfs were tried but did not yield any fruitful result.

5. CONCLUSION

This project has immense scope for development.

1. We can develop algorithms to constantly keep updating the real time costs.
2. We can design specific notifications to be received on the Mobile app, once there is an expected rise of peak consumption on the Utility Server.
3. We can have a common database on the Cloud, to enable faster queries and single database instance, rather than syncing various databases.
4. Tighter integration between different servers.