EE4210 Computer Communication Networks II
Team Project (20% towards module grade)

Team size: two students (please find your fellow team member by 16 Feb 2014 and let us know by sending an email per team to "f.rassaei@nus.edu.sg"). Note that you need to highlight your individual contributions in your final submitted report.

Note that we use IVLE discussion forum to discuss the ambiguities and questions. Please write there your questions, comments, useful references, etc., and join the discussions.

Objectives:
1) Hands-on exercise with socket programming, and multithreaded programming.
2) Experience the design and implementation of an application that resembles a lightweight version of networking in a "Smart Grid" application.
3) Investigate different methods which can manage the power consumption in a grid and analyze their specific communications and networking requirements.

Project Grade:

- Program:
  - Code compilability 15%.
  - Meeting the required features 25%.
  - Comments in the codes 10%.
  - Coding strategy and efficiency 10%.
- Report:
  - Introduction which describes why you have used your particular approach to implement this system 10%.
  - A concise user guide which explains the different steps of how one could use your program and the result of each step 20%.
  - Bugs and limitations of your program and possible debugging schemes 10%.

Problem Description:

The current state-of-the-art in information technology and processing are going to be employed extensively in power grids. The result of such evolution is referred to as a smart grid. Imminent widespread deployment of advanced metering infrastructure (AMI) enables two-way information exchange between consumers and the utility. This changes all different segments of the grid from the generation side to the consumption end.

In fact, one of the main objectives defined for the smart grid is to make the demand responsive. Responsive demand can change the current approach in power dispatching in which supply always follows the demand. Currently, the utility company designs and installs the grid's infrastructure such that it can provide power to adverse daily consumption profiles similar to the one in Fig. 1. This power profile has significant peak-to-average ratio (PAR) which results in an under-utilized grid most of the time. This motivates intensive research on the strategies that can operate the existing power grid more efficiently so that, as a result, more consumers can be accommodated and served without developing new costly power infrastructure.

The system model is represented in Fig. 2. There are different consumers each owning some appliances. We aim to manage their power consumption to have an aggregated power demand profile with less PAR and possibly with a more flat shape. The "cloud" in Fig. 2 represents the communications and networking infrastructure behind this power system which are the main subjects of this project.

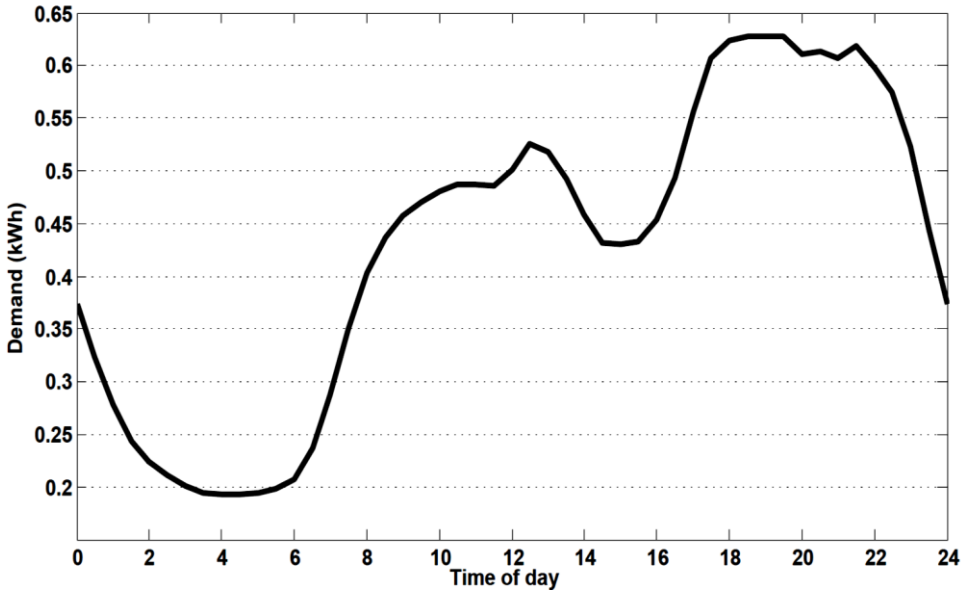

Fig. 1 A typical power demand profile

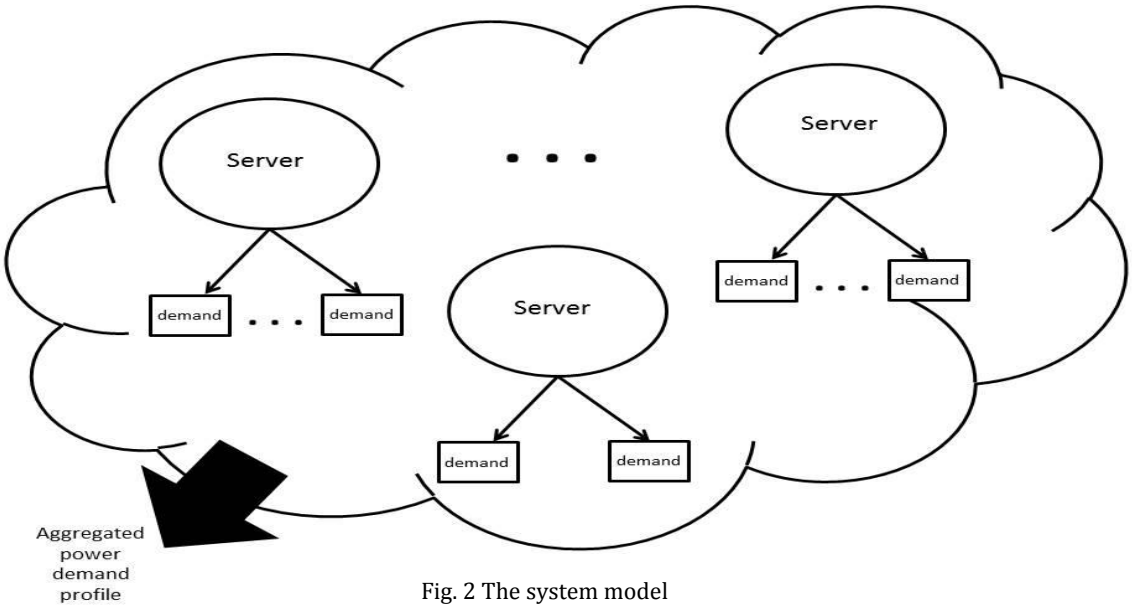

Fig. 2 The system model

You are required to implement a client-server type of application that resembles a lightweight version of this networking, for which you are responsible for writing the program codes for both the client side and the server side. Note that each "server" node in Fig. 2 actually runs both the client side code and the server side code (similar to SMTP email servers that run both client and server codes). The codes for each "server"

node in Fig. 2 must be capable of running on separate machines, and interacting with each other over network connection. But, to simplify the testing of your work, you may implement all the nodes on one machine. The general idea of the application is as follows. When the client side of one node is executed, it will first request a file (ASCII text file) from the server side of another node containing its TOTAL power consumption profile over a 24-hour horizon (a day) with a one-hour resolution (e.g., 12am to 1am: 1kWh, 1am to 2am: 0.8 KWh, …). The client will do this for all other active consumers (servers) in the power system. Upon receiving all of them, the client will try to manage (i.e., adjust) its own power consumption profile accordingly so that the "aggregated power profile of the WHOLE system" achieves some desired characteristics (to be defined below shortly) over that 24-hour horizon.

You are required to implement and investigate the followings:

i. Identify a list of smart residential load appliances whose operating time can be managed, e.g., dishwasher, laundry machines, energy storage devices, electric vehicles (EVs), etc. Define appropriate numbers for their power consumption needs (kWh) in a day.
ii. You need to assume some fixed loads (i.e., not manageable) such as lighting, fridge, TV, heating system, etc. The two lists (manageable and not manageable) should each contain at least 12 different loads.
iii. Consider just three consumers in the power system (for simplicity). These consumers can shift their manageable load throughout the 24-hour horizon. For simplicity, we do not consider any constraint for shifting the "time of use" (e.g., restricting EV charging to nighttime only) of those appliances other than they must receive their total required power according to (i).
iv. During initialization, assign the time of use of the shiftable appliances for different users randomly. You should consider the time granularity (resolution) as 1 hour, i.e., we can just have integers from 0 to 23 as starting times for a job. Note that the appliances will not be turned on and off repeatedly during a day which means they can only be turned on once a day to carry out their jobs.
v. Next, they exchange the information of their resulting total power consumption profile through a network described above.
vi. Upon receiving this information, each consumer tries to shift its own manageable loads with the aim of achieving one of the following objectives:
    - We achieve minimum PAR in a day (24 hour timeframe)
    - We achieve minimum variance of power consumption through a day

    You are required to implement both objectives separately, and compare their final results.
vii. For simplicity, you may use "extensive search" to find the appropriate scheduling (this imposes some limits on the number of your manageable appliances in the whole system). You may also

consider interior point method (IPM) in optimization theory to find the best scheduling (this one gives you bonus points). In either case, you may consider the number of required iterations (the amount of information exchange accordingly) as a design option.

viii. The consumers need to run their search or optimization algorithm individually when it is their turn, i.e., they should not do this simultaneously, but iteratively. Thus, you need to consider an appropriate scheme for scheduling their turns (without assigning any priority to them).

ix. Assume that one connection is lost, i.e., the information about the power consumption profile of one of these three users is not available after several iterations. Observe how this can affect the final result, i.e., compare with the case when no connection is lost.

x. Since each node can only adjust its own manageable loads, it takes multiple iteration rounds before an objective can be achieved globally (i.e., no further iteration is required). Your program must be able to detect this condition and stop the iterations. Your results are the final daily power profile of the whole system and also for each individual user (power consumption vs. time of day), their PARs, and their variances.

Error Handling:

Your program code must be able to handle some common errors, e.g.,

(i) The client and server processes must be capable of gracefully handling the case where the requested file is not present; for instance.

(ii) The server process must not crash or hang if the client to which it is currently sending a file is terminated abruptly (or if the client's network connection is lost).

(iii) The client process must not crash or hang when the server does not provide any file.

Project Submission Deadline:

Due on **23 March 2014 (Sunday)**, **2359 hrs**.

Note that 10% marks (i.e., 2% from your module grade) will be deducted for every day it is late. Hence, you are strongly encouraged to start early (instead of a "slow start").

Deliverables:

Zip all files into a single file, name it using one of your matric numbers, e.g., U0123456A.zip, and upload it into IVLE Workbin under "Project Submission" anytime before the deadline. You must include the following:

(i) Program source codes (not binary executables) for both the client and the server sides. The program codes must be commented properly.

(ii) A written report, in PDF format. You may use your own discretion for the number of pages. Your report must describe your application protocol design, and include the results obtained from the exercises described above. You should also describe the challenges that you'd faced in implementing the project, report any bugs that you cannot solve, and identify the limitations of your design. If you have implemented the extra credit option, please highlight it in a separate section.

(iii) A brief "readme" file (.txt or .doc/.docx) that explains how your code can be compiled, as well as the commands and arguments needed for running your programs. You should also provide some command line examples.
Optional: You may also do a screen video recording with voice narration to demo how your application works. Convert your video into a file format such as .flv which is known to result in smaller file size. Zip your video file together with the rest of your files into a single file as described above. Note that the maximum upload size for IVLE is 200MB.

Academic Integrity:
Plagiarism will not be tolerated. Any form of plagiarism will be reported to the University, and will result in serious consequences.
You are **not** supposed to share any code with any other group. You **may** discuss the project requirements or your solutions, but without sharing details at the code level and the report. If you need the help of a friend to run the server process, you may pass the binary executable file to your friend.

Some Helpful Resources:
You should use either Java, C, or C++ for this project. For any other programming language, please first check with the Teaching Assistant (f.rassaei@nus.edu.sg).

(i) H. Mohsenian-Rad, V. Wong, J. Jatskevich, R. Schober and A. Leon-Garcia "Autonomous demand side management based on game-theoretic energy consumption scheduling for the future smart grid", *IEEE Trans. Smart Grid*, vol. 1, no. 3, pp.320 -331 2010. This paper provides the description of the application in smart grids.

(ii) S. Boyd and L. Vandenberghe *Convex Optimization*, 2004: Cambridge Univ. Press. The interior point method IPM is described in this book.

(iii) Download the following from the IVLE Workbin, under "Project Materials". These are some code examples that we have seen during the lecture. Try them out to familiarize yourself using these simple examples.
- TCPClient.java and TCPServer.java (taken from from Kurose Section 2.7)
- UDPClient.java and UDPServer.java (taken from Kurose Section 2.8)
- WebServer.java (taken from Kurose, 3rd edition, Section 2.9)

(iv) http://www.schaik.com/download/sockets.html, "Web Programming with Sockets". This useful website provides code examples for both Visual C and Java. You will see how a simple Web application (client + server) can be implemented.

(v) http://netbeans.org/index.html, "NetBeans Integrated Development Environment". It is a free, open-source Integrated Development Environment for developing software. It supports Java, C/C++, and more. You are recommended to make use of this tool.

(vi) http://www.netbeans.org/kb/index.html, "Documentation for NetBeans IDE".

(vii) http://java.sun.com/javase/downloads/index.jsp, "Java SE Development Kit".

(viii) http://docs.oracle.com/javase/tutorial/index.html, "The Java Tutorials".

(ix) http://java.sun.com/docs/books/tutorial/getStarted/cupojava/index.html, "Creating "Hello World!" with Java".

Good Luck & Work Hard!