User Monitor & Feedback Mechanism for Social Scientific Study on Laptop Energy Reduction

A Research Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

by

Namrata Buddhadev

Spring 2011

SAN JOSÉ STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

USER MONITOR & FEEDBACK MECHANISM FOR SOCIAL SCIENTIFIC

STUDY ON LAPTOP ENERGY REDUCTION

by

Namrata Buddhadev

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Prof. Mark Stamp, Department of Computer Science Date

Prof. Morris Jones, Department of Electrical Engineering Date

Prof. Robert Chun, Department of Computer Science Date

Abstract

The dawn of the laptop era presents new challenges to the computing research community. Performance is no longer the end all be all of computing research because a new, significant requirement has cropped up: reducing the power consumption and improving battery usage. The key restriction to mobility is the constant need to be near a power outlet to recharge the laptop battery. Thus, extending battery life without compromising performance is a pressing concern.

A research named as "iGreen" was proposed to stimulate power consumption awareness and behavioral change among laptop users via interactive energy-usage feedback with college students as the initial target group.

This project is part of iGreen the project with an objective to analyze the users' behavioral changes by examining their usage of laptops and their attitude towards the environmental issues for energy consumption. It also tries to find reasoning and provide a social feedback mechanism to improve the efficiency of energy consumption and reduce the waste of power.

Acknowledgment

Firstly, I would like to thank to Dr. Mark Stamp & Prof. Morris Jones, my project advisors, for their guidance and support throughout my Masters' degree and project.

Secondly, I am highly indebted to Dean Belle Wei and Prof. Donald Hung for giving me chance to work on the research project. I thank them for the patience, ideas, suggestions and inspirations without which this research project would not have been possible.

Next I would like to thank my committee member, Dr. Robert Chun for helping me during the project and providing his guidance.

Lastly, this acknowledgment would be incomplete without thanking Jay Patel and Pankaj Sitpure for their constant help during research project.

Contents

1. Introduction 6

1.2 Objective of the project 8

1.3 Outline of the project 9

2. Background 10

2.1 Pilot study 11

2.1.1 Observation and results of the pilot study 12

2.1.2 Conclusion of pilot study 20

2.2 Building a new software systems 20

2.2.1 Development of the HCI feedback mechanism 23

3. Collection module 24

3.1 Background 24

3.2 Resource monitor 24

3. 3 Windows task manager 25

3. 5 Implementation 27

3.6 Collection apis 30

4. Transmission module 32

4.1 Bandwidth utilization 32

4.2 Quality of Service (QoS) 32

4.3 Optimal transfer algorithm 33

4.4 Main algorithm 34

4. 5 TCP stream socket server application 38

4.6 TCP stream socket client application 43

5. Encryption: 44

5.1 AES algorithm: 45

5.2 Encryption: 46

5.3 Decryption: 47

6. Test and results 48

7. Conclusion 54

8. Future Work 55

9. References 56

List of Figures

Figure 1: User monitor & feedback mechanism for social scientific study on laptop energy reduction........ 8

Figure 2: Computer operating state........ 13

Figure 3: Samples collected by hour of day........ 14

Figure 4: Plugged/Battery Usage........ 15

Figure 5: Top 9 processes by sample count........ 16

Figure 6: Top 10 processes by CPU consumption (seconds consumed during study period)........ 18

Figure 7: New model........ 22

Figure 8: Resource monitor tool........ 25

Figure 9: Windows task manager........ 26

Figure 10 : Main algorithm........ 35

Figure 11: TCP client - server socket connection........ 40

Figure 12: Socket........ 40

Figure 13: Bind........ 41

Figure 14: AES algorithm [17]........ 45

Figure 15: The server is started and socket is opened for the client to send the data........ 48

Figure 16 : Client........ 50

Figure 17: Received data........ 51

Figure 18: Network utilization........ 52

Figure 19: CPU Utilization........ 53

List of Tables

Table 1. Summary of data collection quantities... 13

Table 2. Top 9 sampled processes 17

Table 3. Top 10 CPU consuming processes by category... 19

List of Code

Code 1: ProcItem 28

Code 2: CPUstuff 29

Code 3: ProcData 29

Code 4: NETstuff 30

Code 5: MIB_IFROW 36

Code 6: MIB_IFTABLE 37

Code 7 : Free network condition 38

Code 8: AES encrypt 46

Code 9: AES decrypt 47

1. Introduction

Energy use related to electronic devices has skyrocketed, especially in the consumer electronics domain [1]. For example, the number of personal computers (PCs) around the world passed the one billion mark in 2008 [2]. The energy costs and related environmental impacts of office computers and networking equipment are well known [3][4][5], but residential and personal use of portable electronic devices also presents a growing concern.

Among all consumer electronic devices, laptop computers being portable and more energy efficient, now account for more than 50% of PCs sold in America. The energy consumption levels of laptops vary depending on their laptop types and user operational modes [6]. Further, user behaviors play an important role in overall laptop energy consumption [7][8].

A research project was proposed by Dean of Engineering, Belle Wei, Prof. Morris Jones, and Prof Donald Hung, at San Jose State University called the "iGreen", to improve the efficiency of laptop energy usage by motivating users to change the way they use their laptops (behavioral change) for efficient energy consumption through persuasive technology. This proposed research will provide laptop designers with realistic feedback about consumer-usage and hence aid the design of improved laptop systems by investigating this area of research by tapping into the college student population, who are avid laptop users. A group of college students was selected because they are avid laptop users with 87.7% of them reporting laptop ownership [25], and user behaviors play an important role in the total laptop system power consumption [26].

In order to motivate user behavioral change, a study was performed to find how college students use their laptops, whether there are correlations between their usage and demographics (e.g., gender), their attitudes toward environmental issues, and to determine effective cognitive and social feedback mechanisms [26] to motivate more energy efficient behavior. The goal of the pilot phase is to explore and develop various components needed to build a laptop usage model, including the software tools for monitoring and collecting laptop usage data, processing and analyzing the collected data, and system-level laptop power measurement. It also helps to sharpen the questions asked of the participants on their attitude towards their relationship with the environment and technology. To obtain laptop system and usage data, a software monitor tool was developed and installed on the participants' laptop, and actual user data was collected.

This research was performed by monitoring the performance of laptop usage using a software monitor tool. The user behavior data was collected, which mainly consisted of information regarding computer resources like CPU, disk read/write, and network traffic.

The first version of the monitor tool collected some network information, which was unnecessary. This project is thus focused on redesigning the monitor tool to collect only the required information and to integrate a transmission module, which will send the collected data to the server for analysis when the network is free.

1.2 Objective of the project

The objective of this project is to redesign the software monitor tool and add more features after the pilot study. This tool collects data from the user's system; constitutes information regarding computer resources like CPU, disk read/write; automatically transmits the data to server for analysis purposes and to implements scientific study on laptop energy reduction by monitoring the user behavior and develop a feedback mechanism for the next experiment which is going to get conducted in Summer 2011 on 100 students. Figure 1 shows the block diagram for the main research proposed, named iGreen. This project is the part of iGreen that implements the collection and transmission module. This new model of the tool will be useful for the next experiment on 100 students to implement the feedback mechanisms for social and scientific study on laptop energy reduction.

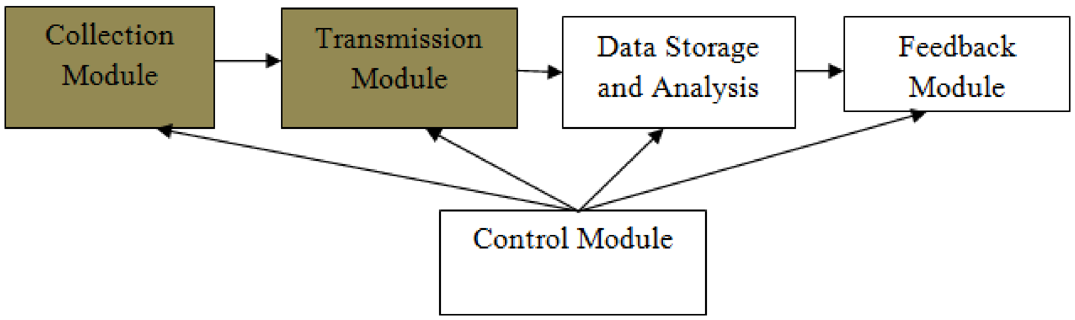


Figure 1: User monitor & feedback mechanism for social scientific study on laptop energy reduction

1.3 Outline of the project

Section 2, begins with the pilot study which was conducted for the iGreen research, which is proposed to stimulate power consumption awareness and behavioral changes among laptop users via interactive energy-usage feedback with college students as the initial target group. Section 3 describes in detail about the background of collection module and its implementation for the monitor tool. In Section 4, the transmission module is covered, which is successfully deployed to transfer the data from client to server. Section 5, discusses AES encryption algorithm, which is implemented to provide the secure data transmission. In Section 6, the tests and results of monitor tool are discussed.

2. Background

For the first time in the history of computing, laptop sales have exceeded desktop sales [6]. The dawn of the laptop era presents new challenges to the computing research community. Performance is no longer the be all end all of computing research because a new, significant requirement has cropped up: reducing the power consumption and improving the usage of battery rather, an important new goal has arisen, improving mobility. The key restriction to mobility is the constant need to be near a power outlet for recharging the laptop battery. Thus, extending battery life without compromising performance becomes a pressing concern.

To achieve this, the first step is to develop a baseline consumer laptop-usage workload, one different from current computer workloads that target the business environment, instead of consumer. Furthermore, exiting benchmarks, especially those in energy efficiency presume simple workloads, fail to reflect the complexity and variety of user behaviors. Examples are Energy Star[4], JouleSort[5], or BAPCo's MobileMark 2007 [6].

Under the research of iGreen project a pilot study was performed on 13 SJSU students. The goal of the pilot phase was to explore and develop various components needed to build a laptop usage workload; including the software tool for monitoring and collecting laptop usage data; processing and analyzing the collected data; and system-level laptop power measurement. It also helps to narrow down specific questions that can be asked to the participants on their attitudes towards their relationship with environment and technology.

2.1 Pilot study

A group of 13 SJSU student participants were recruited via a convenience snowball sampling procedure [34]. Survey data was collected from all 13 participants using the environmental attitudes inventory, a self-report measure of perceptions about the natural environment and policymaking. To obtain laptop system and usage data, a software monitor tool was developed and installed on the participant's laptops, and actual user data was collected for a period of four weeks.

The monitor tool, running in the background without any user interface, used sampling and event-driven mechanisms to gather data on user activities regarding local and web applications every seven to ten seconds. The information collected through the sampling process included the total CPU loading, displaying backlight levels, power source, battery conditions, network interfaces, network cumulative traffic in bytes and packets, as well as all user visible processes and their related resource consumption data. The latter included process creation time, execution time, disk read/write rates, system resources used, and the names of all .exe and .dll files that were loaded. Shutdown events and data on website file types retrieved were also collected

The monitor programs and result data were removed from participants' machines at the end of the study period. That data was transferred to a SQL data base for further analysis.

2.1.1 Observation and results of the pilot study

The data was collected from 13 SJSU students for four weeks. That data was collected via sampling and event-driven mechanisms. The software monitor tool has created a repository with 3 GB of raw data (about 150-250 MB collected from each participant) containing real time laptop usage information. After performing the analysis, it was found that certain information for some known factors that significantly impact laptop power consumption was immediately extracted from the raw data.

Table 1 summarizes some of the information collected from the participants during the study period. The data collection and processing team knew the participants only by number.

Amount	Item
1,067,539	Time point samples
10,792,762	Process sample points
2,862,556	CPU seconds tracked to Processes
128,986	Process to DLL bindings
832	Unique processes
461,572	Input intervals
5,122,169	Network interface items

385	Shutdown events
3	Operating system releases

Table 1. Summary of data collection quantities

Figure 2 shows the aggregated time distribution for different laptop operation states in terms of shut (off), sleep, and active.

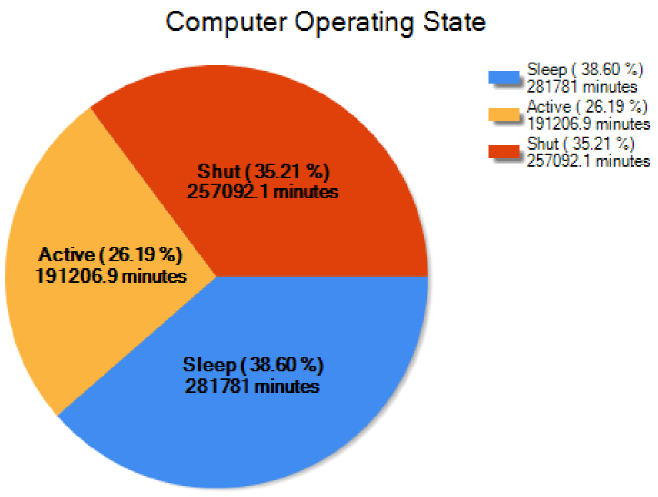


Figure 2: Computer operating state

Figure 3 shows the number of samples collected by time of day during the study period. Samples were collected whenever the machine was running. This was used as an indicator of the amount of time the user has the machine active or idle.

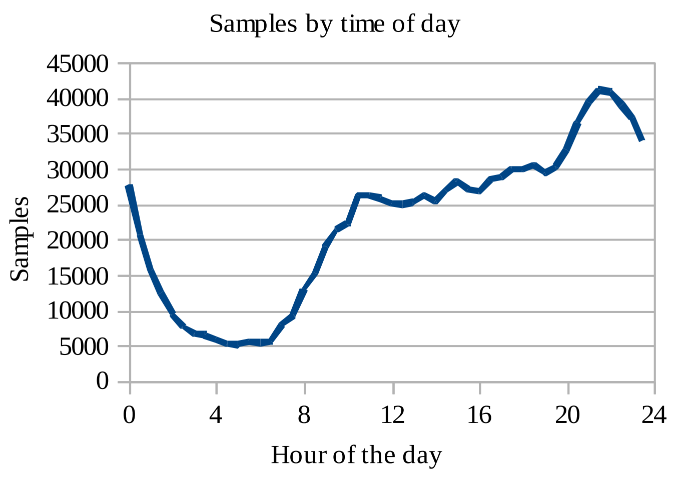


Figure 3: Samples collected by hour of day

It was found that the peak numbers of machines were powered on between 8PM and midnight. The data showed limited student use in the morning. Several of the participants in the study never turned the notebook off. The notebooks ran with the lid closed and they kept charging throughout the day. There were 385 shutdown events recorded in the study. This was about once per day for each study participant.

Battery power

The pilot study results included information about AC/Battery mode, the operation state (off, sleep, idle or active), and brightness levels of backlight displays. It was found that the participants' laptops were running on average 27.8% of the day (6.5 hours/day). While running, they are plugged to the AC power outlet on average 83% of the time [Figure 4]. All of the pilot-study participants' laptops ran with a high-performance, high-power profile while on AC mode.

The energy-consumption difference between AC and battery modes for a laptop running the same application could be significant. For instance, our laboratory power measurement showed it to be 26% (higher for AC) for running a 40-minute DVD video. This difference is attributed to the setting of the laptop's power management profile to low power while on battery. The higher power consumption for the laptop on AC coupled with 83% AC mode operation in the pilot group suggests a fruitful area for energy savings for both users and designers.

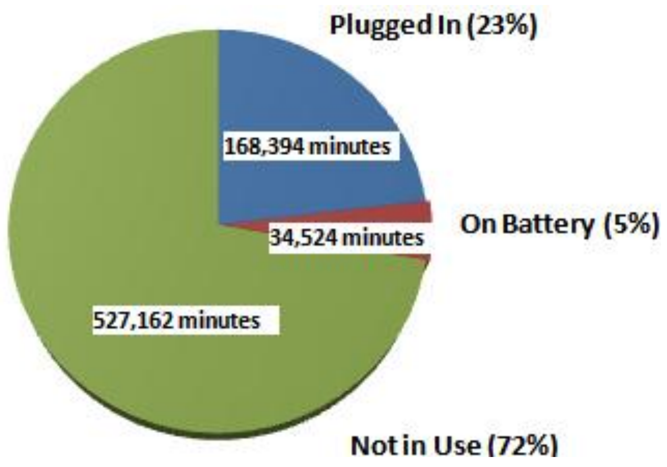


Figure 4: Plugged/Battery Usage

For users, this means there is the possibility of setting the laptop's power management profile to low power, even though it is on AC or switching its mode from AC to battery. For designers, this means there is the possibility for a lower power design for the laptop on AC mode, even though the battery life is not a design concern for this mode. Pilot data showed that, while on AC, the automatic sleep or hibernate mechanism was typically disabled in the participants' laptops. Convincing users to change their laptop operation states (sleep/hibernate/shutdown) while on AC can result in significant energy savings [33].

Processes in use

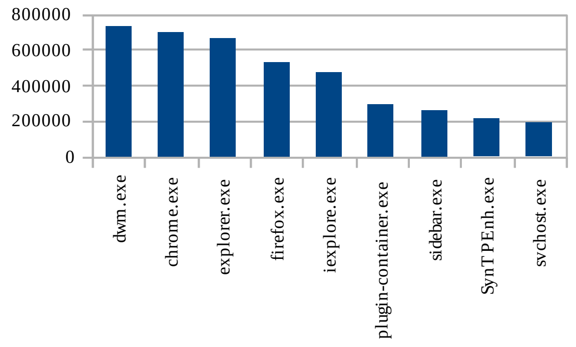


Figure 5: Top 9 processes by sample count

The system collected information on user visible processes per each sample. On average, there were 10 visible processes used by the participant's machine per each sample. Five of the top 10 user visible processes are related to operating system functions. The frequency of sighting is shown in figure 6 for the top 9 sighted processes. Table 2 identifies user application processes.

Name	**User/OS**	**Description**
Dwm.exe	OS	Display manager
Chrome.exe	User	Web browser
Explorer.exe	OS	User shell
Firefox.exe	User	Web browser
Iexplore.exe	User	Web browser
Plugin-Container.exe	User	Second process for Firefox. (Runs flash typically)
Sidebar.exe	OS	User shell application

SynTPEnh.exe	OS	Notebook touch pad interface
Svchost.exe	OS	Dynamic linking support

Table 2. Top 9 sampled processes

These numbers do not show the amount of resources consumed, just the amount of time the application process was present. The sum of the three web browser samples was about 2.5 times greater than the number of user shells. This implies that at any one time, a typical user had 2.5 browser applications open. Each browser application had multiple tabs or sessions active within the process.

The system collected the CPU times for each process and sample. Cumulative values were recorded, and the resources consumed in each sample period were computed by the differences from a previous period. Figure 7 shows the top 10 processes by user CPU consumption.

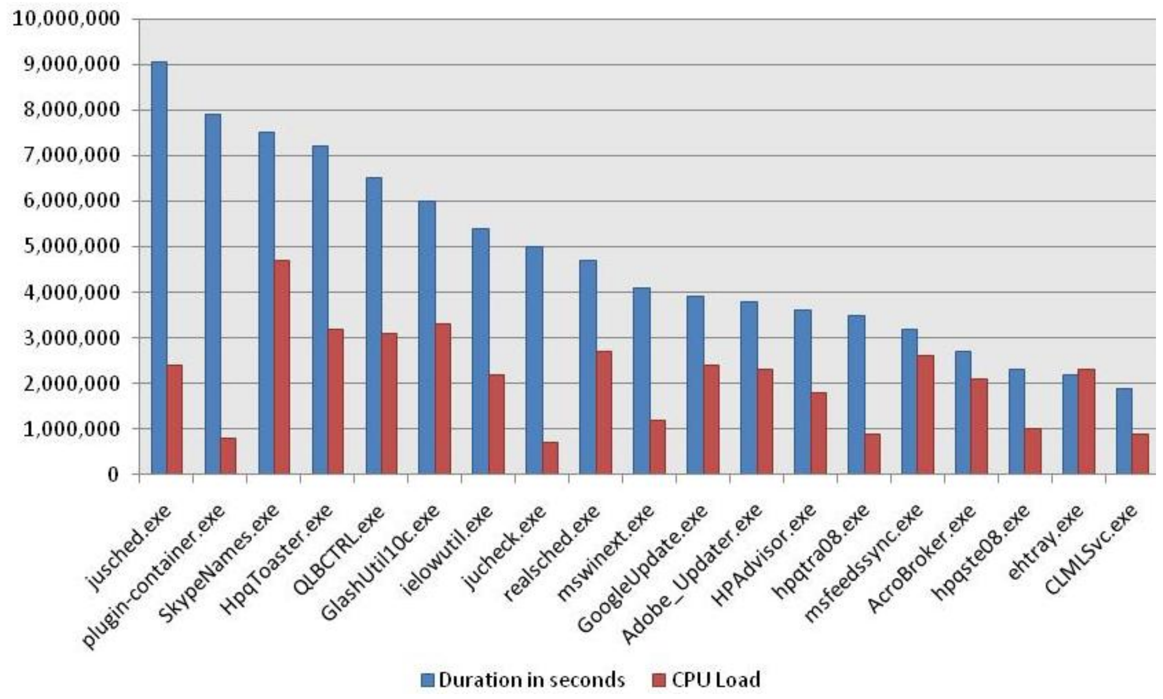


Figure 6: Top 10 processes by CPU consumption (seconds consumed during study period)

The applications were analyzed and placed in a set of broad categories. This is shown in table 3. The system reported both user and kernel times allocated to the application processes. It was observed that some applications had more kernel time than user time. There were 2.2M CPU seconds of user times reported, and 608K seconds of kernel times. This is an area for further study.

Category	**Total user CPU seconds**
Browsers • firefox.exe • chrome.exe • plugin-container.exe • iexplore.exe	1,217,000
Media • convertXtoDVD.exe	373,000

• wmplayer.exe • ViiKiiDesktopPlugin.exe	
Games • java.exe • AttackonPearlHarbor.exe	97,000
Operating System	89,000

Table 3. Top 10 CPU consuming processes by category

Browsers consumed over half of the total CPU times of the study. Outside of these groupings, all of the other applications consumed approximately 25% of the CPU resources during the study period. The notebooks appeared appear to be used primarily for web type applications. Word processing is number 17 in the list of CPU consumption, and just beat instant messenger. Total word processing consumed 28,898 seconds CPU time. One participant consumed more CPU time playing a game than the entire study group consumed during word processing. Students may justify portable computers for educational purposes, but they appear to use them primarily for browsing, media applications and games.

Background applications are considered to be the main area of research. Indeed, the application that used most CPU resources in the pilot study was a background application. As our pilot study demonstrated, a significant amount of CPU resources were consumed by background applications, which include messengers, application updaters, VoIP (Voice over IP) listeners, and screen effects such as information feeds and photo albums that constantly change [Figure 4]. These applications tend to be installed with software packages; the user seldom installs them directly. Behavioral changes could include changing the auto start settings of these applications.

2.1.2 Conclusion of pilot study

The pilot study for the iGreen project was completed. As the project iGreen continued the recorded usage data was aggregated and represented by a set of attributes. Each represented a certain type of activity with its energy consumption information obtained from laboratory measurement. Since the recorded usage data was time stamped, the different usage patterns (e.g., time spent on a particular type of activity, with the usage intensity indicated by numbers of overlapped attributes in the same time interval) of each user were extracted and presented in the form of pattern vectors. Furthermore, statistical pattern classification techniques were applied on a large set of user samples to form a patterned space. This patterned space consisted of various classes of usage patterns with known energy consumption levels, and to developed a pattern classification mechanisms [28],[29]. Based on which, the energy-use feedback can be provided to the user.

A new study of a group of over 100 students is scheduled for this summer using a newly developed software system with a transmission module which automatically transfers client data to server. After performing data analysis on the server, the user feedback is provided for the purpose of reducing laptop energy.

2.2 Building a new software systems

In preparation for the next experimental phase, software systems will be developed to implement the feedback mechanisms designed by the behavioral science team. They will monitor user behaviors, track behavioral changes, as well as the infrastructure, to ensure the privacy and security of user data that will be transmitted and stored.

In new collection process, the participants' data will be transported to a server over a public network using three layers of encryption. They will data will first be encrypted using the public key encryption of AES cryptography. This public key system provides user protection, and ensures proper participant identification. This is done using openssl libraries. The data will be decrypted on the server and placed in SQL (Structured Query Language) data base for analysis.

The new plan is proposed to have two servers, one secure primary server and a secondary server. The primary server will process data for general dissemination from the raw data and place it them on the secondary server. The secondary server will have no participant identifiable data.

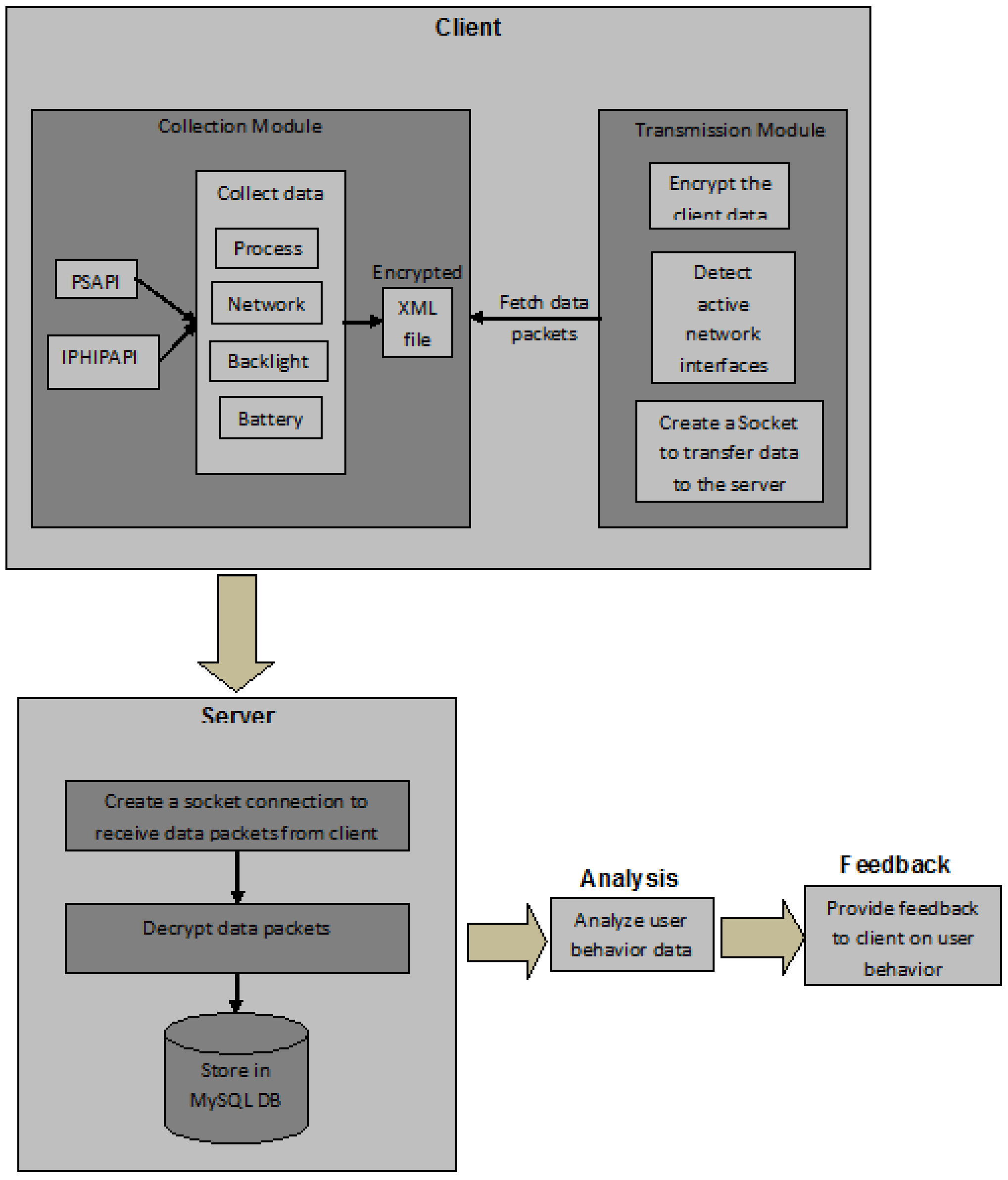


Figure 7: New model

2.2.1 Development of the HCI feedback mechanism

Working with the engineering team, the behavioral science team will develop an energy-use feedback mechanism with a compelling user interface design which will effectively educate and motivate users to improve their laptop use for better energy efficiency. The feedback mechanism monitors the state of targeted energy conservation acts and initiates a trigger on the screen of a laptop, once the state exceeds a threshold energy profile. While we might expect the participants to be self motivated to respond to the trigger to some extent, our interface design informed by the HCI (human-computer interaction) literature will incorporate visualization [10], feedback [11][12][13], discovery and motivation design techniques [14], and easy actionable steps [15][16] to make a significant difference (i.e., to strengthen motivation and simplify a needed ability enough for laptop users to take targeted action more successfully).

Our objective is to design a dynamic and compelling visual user interface that motivates participants to take actions, implicitly educating them towards adopting a more self-directed energy saving behavior in the long run. When a user further interacts with the interface, s/he will discover more details about the specific user-controlled behaviors. The machine will state the underlying the energy use indicator, as well as motivating patterns and trends (perhaps with competitive elements). Once the user responds to a specific "trigger" by taking the corresponding action, its underlying energy profile will change to a lower state and the trigger will then disappear from the laptop screen.

3. Collection module

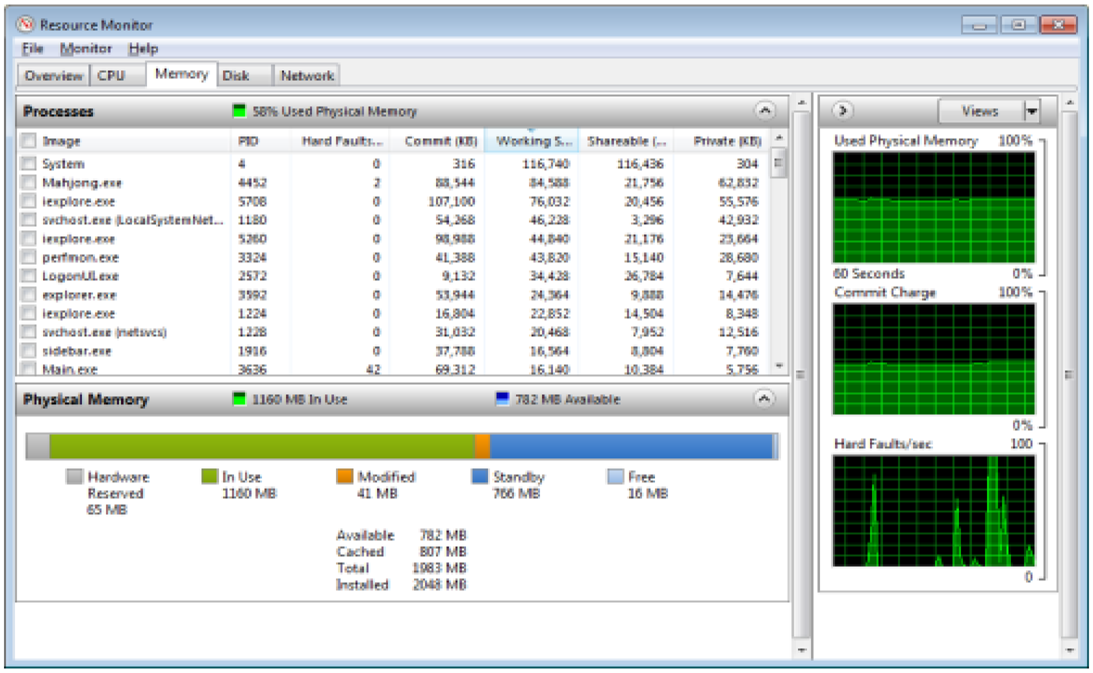
The collection module is a component of the monitor tool that assembles an array of statistics to measure the user’s behavior.

3.1 Background

A Microsoft tool known as the Resource Monitor performs a similar function like monitor tool.

3.2 Resource monitor

The Windows Resource Monitor is a tool for understanding how your system resources are used by processes and services. In addition to monitoring resource usage in real time, the Resource Monitor can help you analyze unresponsive processes; identify which applications are using files, and control processes and services. [22] It gathers information regarding CPU, Memory, Disk, and Network.



Processes 58% Used Physical Memory

Image	PID	Hard Faults...	Commit (KB)	Working S...	Shareable (...	Private (KB)
System	4	0	316	116,740	116,436	304
Mahjong.exe	4452	2	88,544	84,588	21,756	62,832
iexplore.exe	5708	0	107,100	76,032	20,456	55,576
svchost.exe (LocalSystemNet...	1180	0	54,268	46,228	3,296	42,932
iexplore.exe	5260	0	98,988	44,840	21,176	23,664
perfmon.exe	3324	0	41,388	43,820	15,140	28,680
LogonUI.exe	2572	0	9,132	34,428	26,784	7,644
explorer.exe	3592	0	53,944	24,364	9,888	14,476
iexplore.exe	1224	0	16,804	22,852	14,504	8,348
svchost.exe (netsvcs)	1228	0	31,032	20,468	7,952	12,516
sidebar.exe	1916	0	37,788	16,564	8,804	7,760
Main.exe	3636	42	69,312	16,140	10,384	5,756

Physical Memory 1160 MB In Use 782 MB Available

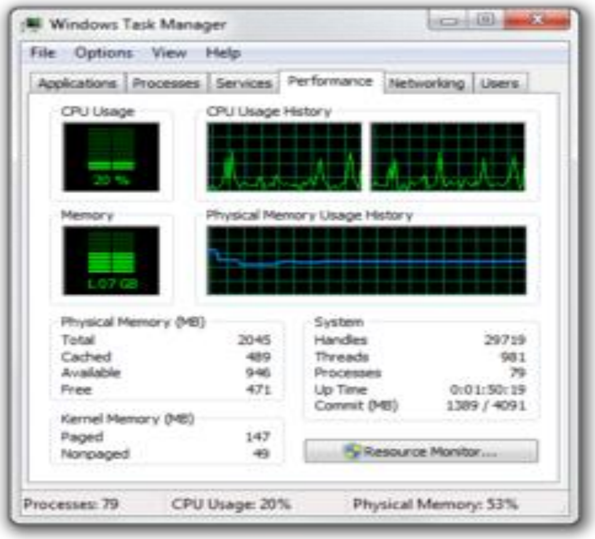
Hardware Reserved 65 MB	In Use 1160 MB	Modified 41 MB	Standby 766 MB	Free 16 MB

Available	782 MB
Cached	807 MB
Total	1983 MB
Installed	2048 MB

Figure 8: Resource monitor tool

3. 3 Windows task manager

Windows has another tool known as task manager, which is available in all versions of windows starting from Windows NT. It provides detailed information regarding the computer performance like CPU usage, commit charge and memory information, network activity and statistics, logged-in users, and system services. It has the ability to set priorities by a processor affinity along with the functionality of forcibly terminating the processes and shut down, hibernate and log off from windows [24].



Physical Memory (MB)
Total 2045
Cached 489
Available 946
Free 471

Kernel Memory (MB)
Paged 147
Nonpaged 49

System
Handles 29719
Threads 981
Processes 79
Up Time 0:01:50:19
Commit (MB) 1389 / 4091

Processes: 79 CPU Usage: 20% Physical Memory: 53%

Figure 9: Windows task manager

There are a few reasons why we needed to develop the monitor tool even though there are other tools with a similar functionality, like resource monitor and task manager, which are a utility in windows:

- The Resource monitor utility for Windows is only present in Windows 7 and Vista, while it is not present in Windows XP.
- The Resource monitor tool and task manager cannot log the data, which can be further analyzed to check the behavior of the user.
- The Resource monitor is developed to provide feedback to the user regarding his/her behavior and to reduce power waste– a functionality which is not available in any of the tools like resource monitor and task manager.

3. 5 Implementation

To obtain a laptop system and usage data, a software monitor tool was been developed and installed on the participants' laptops. The actual user data was been collected for a period of four weeks during the pilot study. The monitor software, running in the background without any user interface, used two mechanisms for data collection: sampling and event driven. The sampling mechanism initiated a collection process that gathered information about user and system activities following a 5-second pause. The collection process took two to five seconds to complete due to system scheduling of operating system calls. The sampling period was then about seven to ten seconds. The information that was collected in this way included the total CPU loading, displayed backlight levels, power source, battery conditions, network interfaces, network cumulative traffic in bytes and packets, as well as all user visible processes and their related information. The process related information included creation time, execution time, system resources used, and the names of all *.exe* and *.dll* files loaded. Shutdown events are also collected. Data collection based on event driven mechanism was activated by a network request with a destination of port 80 and "GET" at the beginning of the message. It then monitored web-based traffic requests, and collects only the file type information. This data, together with the network cumulative traffic data, provided information on the web traffic flow.

The following code sections give information about the variety of data collected using the collection routine of this monitor tool.

```
class ProcItem
{
public:
      ProcItem *fpt;                  // forward list pointer.
      char *Name;                     // The name of the process
      FILETIME CreationTime;    // When the process was started
      FILETIME ExitTime;              // When the process ended
      FILETIME KernelTime;      // Amount of CPU time in the kernel
      FILETIME UserTime;              // Amount of CPU time allocated to the User
      IO_COUNTERS io_counters;// The amount of IO performed
      DWORD GDI_objects;              // The number of GDI objects allocated
      DWORD Peak_GDI_objects;   // The peak number of GDI objects allocated
      DWORD User_objects;             // The number of user objects allocated
      DWORD Peak_User_objects;// The peak number of User objects allocated
      DWORD PriorityClass;      // The priority class number for this item
      DWORD HandleCount;              // The number of handles this process has
allocated
      ProcMod *pm,*pmt;         // list of the module names in the process



      ProcItem(void);
      ~ProcItem(void);
      void addName(WORD *wname);
      void addModule(WORD *mname);
};
```

Code 1: ProcItem

```
class CPUstuff
{
public:
	CPUstuff *fpt;			// forward list pointer
	char *Name;			// The name of this CPU
	DWORD clockSpeed;	// the current clock speed
	DWORD loadPercentage;	// the percentage this CPU is loaded
	DWORD currentVoltage;	// the current voltage times 10 of the
CPU
	DWORD numberOfCores;	// The number of cores in the CPU chip
	DWORD numberOfLogicalProcessors;	// The number of logical
processors
								//
(different than the number of cores if hyperthreading)
	CPUstuff(WORD *name);
	~CPUstuff();
};
```

Code 2: CPUstuff

```
class ProcData
{
public:
	SYSTEMTIME snapshotTime;
	SYSTEMTIME lastshotTime;
	ProcItem *piList,*piTail;
	DWORD ACLineStatus;
	DWORD BatteryFlag;
	DWORD BatteryLifePercent;
	DWORD BatteryLifeTime;
	DWORD BatteryFullLifeTime;
	FILETIME SIdleTime;
	FILETIME SKernelTime;
	FILETIME SUserTime;
	DWORD TicCount,LastInput,TicIncrement;
	DWORD ACbright,DCbright;
	DWORD NumberOfProcessors;
	CPUstuff *CPUhead,*CPUtail;

	ProcData(void);
	~ProcData(void);
	ProcItem *addProc(WORD *name);
	//NETstuff *NEThead,*NETtail;
	void addNET(NETstuff *ns);
	void addCPU(CPUstuff *cs);
};
```

Code 3: ProcData

```
class NETstuff
{
public:             // no real security here...
      NETstuff *fpt;
      char Name[500+10];
      char Desc[500];
      DWORD type,speed,opStatus,adminStatus;
      DWORD oIn,oOut;
      DWORD uPin,nuPin;
      DWORD uPout,nuPout;
      NETstuff();
};
```

Code 4: NETstuff

The above code sections reflect various types of data collected using different Apis. This Apis can be summarized as follows:

3.6 Collection apis

Iphlpapi.h

This header file is implemented in the msdn library. It provides functionality to collect the information related to network interface; such as the name of network interface, type of network interface, network speed, operational status, unicast packets in & out and non -unicast packets in & out. [31]

Psapi.h

The process status application programming interface (PSAPI) is a helper api implemented in the msdn library- which gives process information like: device drivers process creation time, ending time, user time, kernel time, its GDI object details, handle count and process module details.

The battery information like AC indicator flag, battery flag, and battery life time is obtained from the power status class in “Windows.Forms” namespace.

The System information like system idle time, kernel time, tick interval, last input time, display AC & DC brightness along with CPU information like number of cores and number of processors is also gathered using the monitor tool [30].

4. Transmission module

Transmission module is a component implemented in the monitor tool which acquires the data from the collection module deployed on the client side and sends it over to database on server side.

4.1 Bandwidth utilization

Bandwidth is the primary measure of computer network speed. It represents the capacity of the connection. The greater the bandwidth the better the network performs. Bandwidth is the amount of data that passes through a network connection over time as measured in bps. It refers to the data rate supported by a network connection or interface. It is not the only factor that contributes to the network speed. Another key element for the network performance is latency. The latency refers to the several kinds of delays typically incurred while processing the network data [18].

4.2 Quality of Service (QoS)

Quality of service is the ability to give priorities to different applications and users. It is to guarantee a certain level of performance to a data flow. Hence, a required amount of bit rate, delay, jitter, packet dropping probability and/or bit error rate may be guaranteed. QoS also guarantees network capacity for real time streaming multimedia applications that require fixed bit rate and are delay sensitive, and for cellular data communication.

The network protocol, which supports QoS provides an opportunity for traffic contract with the software application by reserving capacity in the network for each node during the session establishment phase. It may monitor the performance during the session and calculate the data rate. It may also delay the performance by dynamically controlling and scheduling the priorities in the network nodes. Thus, QoS may release the reserved capacity in the tear down phase.

QoS cannot be achieved by the best–effort network service, however the alternative to complex QoS is high quality communication over a best effort network by over provisioning the capacity to achieve sufficient peak traffic load. Thus, by over provisioning the capacity, we can get rid of network congestion and therefore eliminate the need of QoS mechanism [21].

4.3 Optimal transfer algorithm

The optimal algorithm provides following facility,

- To fully utilize the communication bandwidth,
- To preset the bandwidth consumption on a needed level,
- And helps to deal with the situation, where any client unnecessarily requests too high of an update rate [20].

There are possibilities of two types of disadvantages in the client-server connection when the application reads or writes the data in different process variables. They can be summarized as follows:

1. Clients need to communicate according to the priority of the process.
2. Different clients send the data to the server at a time in whole communication path.

To reduce communication overhead, real time process data needs to be read in blocks. These blocks can be grouped. Each group can be assigned a sampling rate and can be scheduled in the sampling queue as one item. This sampling rate is a function of client requests and limitations set in the server configuration. The scheduling algorithm minimizes the difference between the real time sampling rates to the evaluated sampling rate [20].

There are few factors to consider while the data transfers on the network like: transfer scheduling, resource reservation for guaranteed-bandwidth, advance reservation and time constraints.

After considering the above factors, an algorithm was implemented to transfer the data collected using the collection module of the monitor tool, from client to server when the client network is free.

4.4 Main algorithm

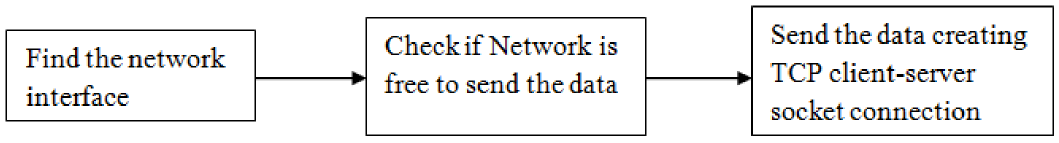

Find the network interface
Check if Network is free to send the data
Send the data creating TCP client-server socket connection

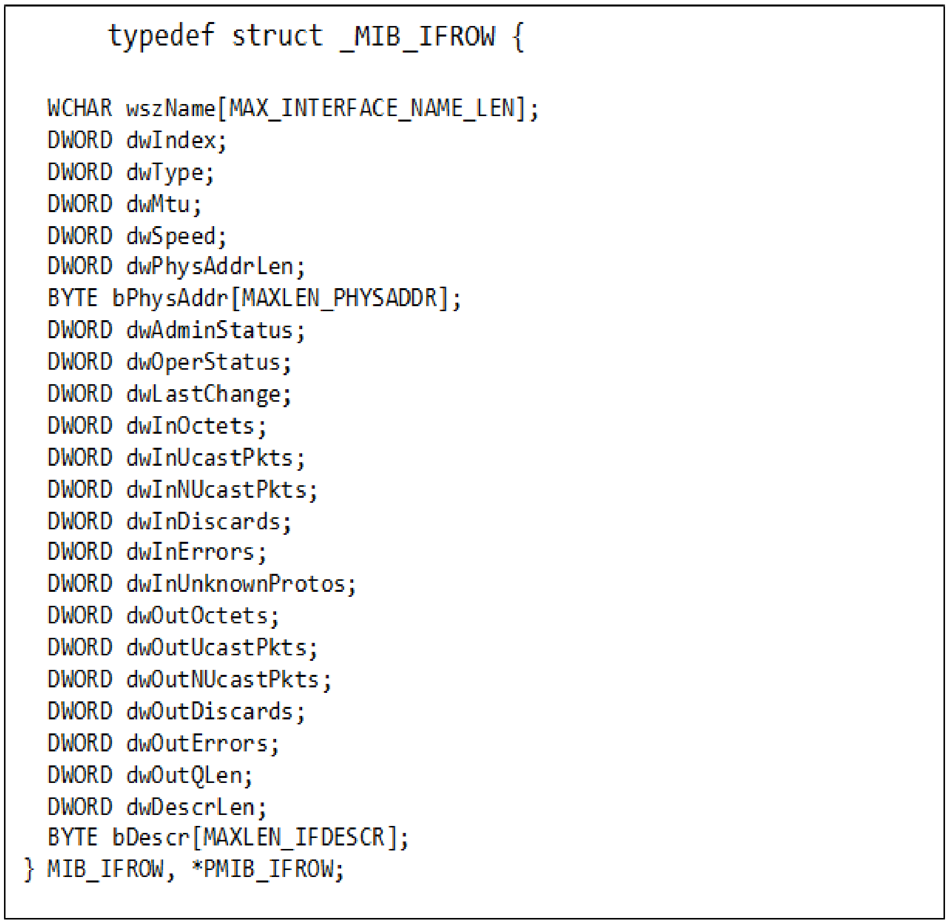
Figure 10 : Main algorithm

1) To find the network interface:

The network interface details are obtained from the two major functions **MIB_IFROW** and **MIB_IFTABLE** present in the msdn library like winsock2.h and iphlpapi.h to find the details about the network interface.

The following structure of MIB_IFROW is used to find the detailed information about particular interface

- The **MIB_IFROW** structure stores information about a particular interface.



```
     typedef struct _MIB_IFROW {

  WCHAR wszName[MAX_INTERFACE_NAME_LEN];
  DWORD dwIndex;
  DWORD dwType;
  DWORD dwMtu;
  DWORD dwSpeed;
  DWORD dwPhysAddrLen;
  BYTE bPhysAddr[MAXLEN_PHYSADDR];
  DWORD dwAdminStatus;
  DWORD dwOperStatus;
  DWORD dwLastChange;
  DWORD dwInOctets;
  DWORD dwInUcastPkts;
  DWORD dwInNUcastPkts;
  DWORD dwInDiscards;
  DWORD dwInErrors;
  DWORD dwInUnknownProtos;
  DWORD dwOutOctets;
  DWORD dwOutUcastPkts;
  DWORD dwOutNUcastPkts;
  DWORD dwOutDiscards;
  DWORD dwOutErrors;
  DWORD dwOutQLen;
  DWORD dwDescrLen;
  BYTE bDescr[MAXLEN_IFDESCR];
} MIB_IFROW, *PMIB_IFROW;
```

Code 5: MIB_IFROW

- The **MIB_IFTABLE** structure contains a table of interface entries, which were connected and used by the user.

```
typedef struct _MIB_IFTABLE {
  DWORD     dwNumEntries;
  MIB_IFROW table[ANY_SIZE];
} MIB_IFTABLE, *PMIB_IFTABLE;
```

Code 6: MIB_IFTABLE

2) Check if Network is free to send the data:

The network data can be transferred from the client to the server when the client side network is free. To check the free network, the IN & OUT discards and IN & OUT errors are considered.

In Errors & Out Errors

Type: DWORD

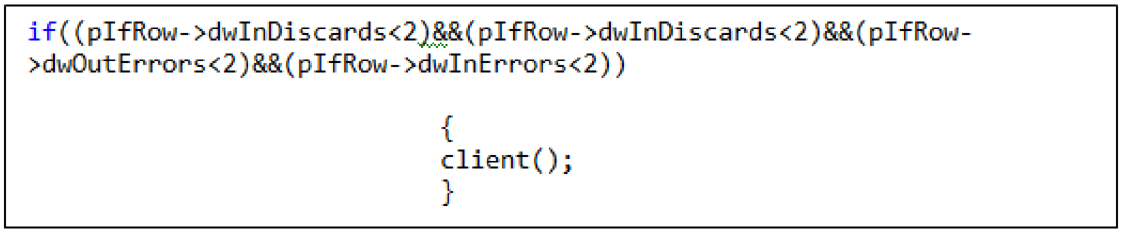
The number of incoming and outgoing packets that were discarded because of errors.

In Discards & Out Discards

Type: DWORD

The number of incoming and outgoing packets that were discarded even though they did not have errors.

The above two factors will help to find if the client network is free enough to send the data to the server. So, if following condition satisfies, then we can consider that the client network is available for sending the collected data from the collection module of the monitor tool.



```
if((pIfRow->dwInDiscards<2)&&(pIfRow->dwInDiscards<2)&&(pIfRow->dwOutErrors<2)&&(pIfRow->dwInErrors<2))

                              {
                              client();
                              }
```

Code 7 : Free network condition

In & out discards along with In & Out errors are less than 2.

3.) Create TCP socket connection described in the next section.

4. 5 TCP stream socket server application

The sockets are a protocol to create a connection between the processes. It can create either a connection based or connectionless connection. The main socket characteristic is AF_INET, which provides the format of a host and port number. The connection based socket communicates between client and server, wherein the server waits for a connection from the client.

The main api’s used for establishing the connection are as follows:

- **Socket:** To create a socket of particular type.
- **Bind:** It allocates a name to the socket.
- **Listen:** It specifies the number of pending connections that can be queued for a server socket.
- **Accept**: server accepts a connection request from a client.
- **Connect:** This API is to create a client request for connection with server.
- **Send:** Write to connection
- **Recv**: Read from connection
- **Shutdown:** End of the call

The following block diagram helps to understand the steps to establish the client server connection: [23]

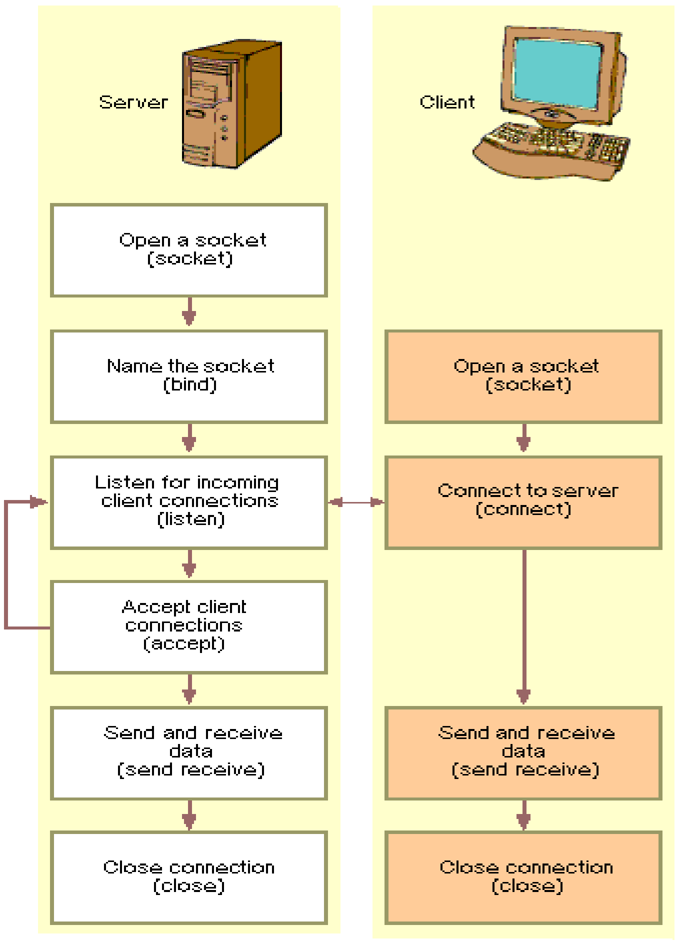


Figure 11: TCP client - server socket connection

1) **The first step is to open the server socket by using AF_INET for the address format and SOCK_STREAM**

```
if ((WinSocket = socket (AF_INET, SOCK_STREAM, 0)) ==
INVALID_SOCKET)
{
  wsprintf (szError, TEXT("Allocating socket failed.
Error: %d"),
            WSAGetLastError ());
    return FALSE;
}
```

Figure 12: Socket

2) The next step is to name the socket with the bind function using a SOCKADDR_IN structure for the *address* parameter.

The bind function is called to assign a name to the server socket.

The bind helps to establish the socket association. The following code example shows how to initialize **SOCKADDR_IN**, and then use **bind**.

```
//Fill out the local socket address data.
local_sin.sin_family = AF_INET;
local_sin.sin_port = htons (PORTNUM);
local_sin.sin_addr.s_addr = htonl (INADDR_ANY);

// Associate the local address with WinSocket.
if (bind (WinSocket,
          (struct sockaddr *) &local_sin,
          sizeof (local_sin)) == SOCKET_ERROR)
{
  wsprintf (szError, TEXT("Binding socket failed. Error:
%d"),
            WSAGetLastError ());
  MessageBox (NULL, szError, TEXT("Error"), MB_OK);
  closesocket (WinSocket);
  return FALSE;
}
```

Figure 13: Bind

3) Listen for incoming client connections with **listen.**

To prepare for a name association, the TCP stream server must first listen for connection requests from the TCP client with the **listen** function.

```
if (listen (WinSocket, MAX_PENDING_CONNECTS) == SOCKET_ERROR)
```

4) Accept a client connection with the **accept** function.

The tcp stream socket uses **accept** to accept the client connection that completes the connection. [23]

The following code example shows how to use **accept**.

```
ssock = accept(msock, (struct sockaddr *)&fsin, &alen);
```

5) Send and receive data with a client using the **send** and **recv** functions.

After the client server socket gets connected send and recv functions can be used for exchanging the data.

```
recv (ClientSock, szServerA, sizeof (szServerA), 0);


send (ClientSock, "To Client.", strlen ("To Client.") + 1, 0)
```

However, successfully completing a call to **send** does not confirm that data was successfully delivered.

6) Close the connection with the **closesocke**t function. [23]

4.6 TCP stream socket client application

The following steps list how to create a TCP stream socket client connection with the server:

1) Open a stream socket using AF_INET.
2) Connect to the server using a **SOCKADDR_IN** structure.
3) The TCP stream client associates socket names by connecting to the stream server with **connect**.

   The following code example shows the TCP client connecting with the server.

   ```
   // Establish a connection to the server socket.
   if (connect (ServerSock,
               (PSOCKADDR) &destination_sin,
               sizeof (destination_sin)) == SOCKET_ERROR)
   ```

4) Exchange data with a server, using the **send** and **recv** functions.
5) Close the connection with **closesocket** [23].

5. Encryption:

To make the connection more secure, data is encrypted before sending to the server from client side. AES is a standard cryptographic algorithm wherein a symmetric key encryption standard is adapted. This standard consists of three block ciphers as: AES-128, AES-192 and AES-256. Each of the cipher has a key size of 128, 192, and 256 bits, respectively. The AES algorithm executes number of repetitions in order to transform plaintext to cipher text. Each round comprises of following steps:

1) Key Expansion step is derived from the cipher key using Rijndeal's schedule.
2) In the Initial round, each byte is combined with the round key using bitwise XOR.
3) In the Rounds step, various steps like adding SubBytes according to the look up table, shifting rows, mixing columns and adding AddRoundKey are performed.
4) Final Round: In final round above steps of adding subtypes, shifting rows and AddRoundKey are repeated without mixing the columns.

5.1 AES algorithm:

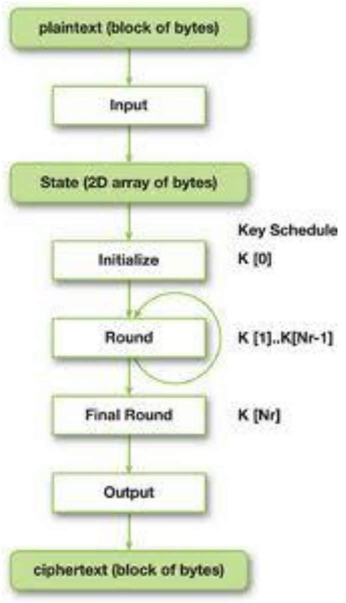

plaintext (block of bytes)
Input
State (2D array of bytes)
Key Schedule
Initialize
K [0]
Round
K [1]..K[Nr-1]
Final Round
K [Nr]
Output
ciphertext (block of bytes)

Figure 14: AES algorithm [17]

To perform the AES encryption, **evp.h, openssl** libraries is used to implement the following main function of encryption and decryption.

5.2 Encryption:

```
unsigned char *aes_encrypt(EVP_CIPHER_CTX *e, unsigned char *plaintext,
int *len)
{
  /* max ciphertext len for a n bytes of plaintext is n +
AES_BLOCK_SIZE -1 bytes */
  int c_len = *len + AES_BLOCK_SIZE, f_len = 0;
  unsigned char *ciphertext = malloc(c_len);

  /* allows reusing of 'e' for multiple encryption cycles */
  EVP_EncryptInit_ex(e, NULL, NULL, NULL, NULL);

  /* update ciphertext, c_len is filled with the length of ciphertext
generated,
    *len is the size of plaintext in bytes */
  EVP_EncryptUpdate(e, ciphertext, &c_len, plaintext, *len);

  /* update ciphertext with the final remaining bytes */
  EVP_EncryptFinal_ex(e, ciphertext+c_len, &f_len);

  *len = c_len + f_len;
  return ciphertext;
}
```

Code 8: AES encrypt

5.3 Decryption:

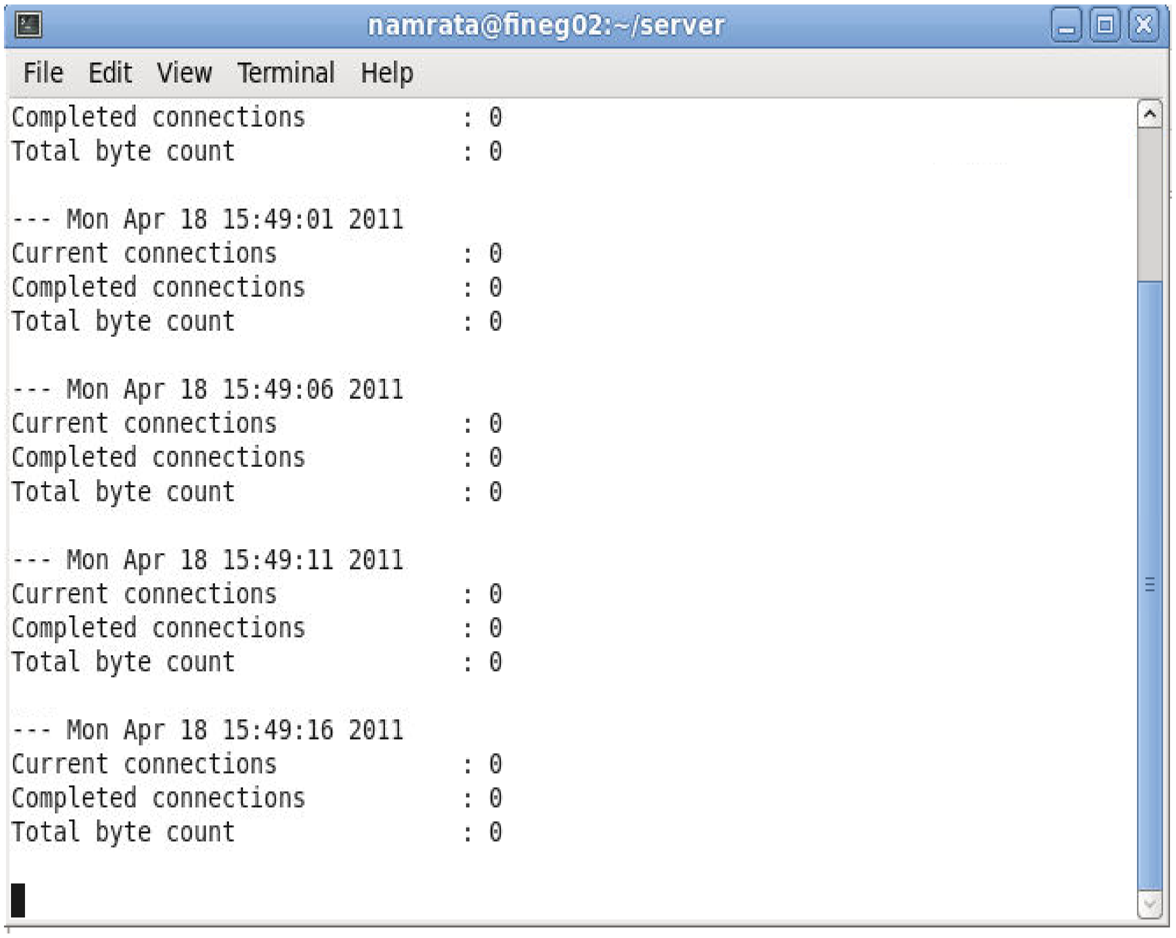
```
unsigned char *aes_decrypt(EVP_CIPHER_CTX *e, unsigned char
*ciphertext, int *len)
{
  /* plaintext will always be equal to or lesser than length of
ciphertext*/
  int p_len = *len, f_len = 0;
  unsigned char *plaintext = malloc(p_len);

  EVP_DecryptInit_ex(e, NULL, NULL, NULL, NULL);
  EVP_DecryptUpdate(e, plaintext, &p_len, ciphertext, *len);
  EVP_DecryptFinal_ex(e, plaintext+p_len, &f_len);

  *len = p_len + f_len;
  return plaintext;
}
```

Code 9: AES decrypt

6. Test and results

A secure client server connection is established. The AES standard cryptography algorithm is used to protect the data transferred from client to server using the TCP socket connection. The following shows the results obtained performing test on the tool:

1) The snapshot below shows the initialized server socket connection.



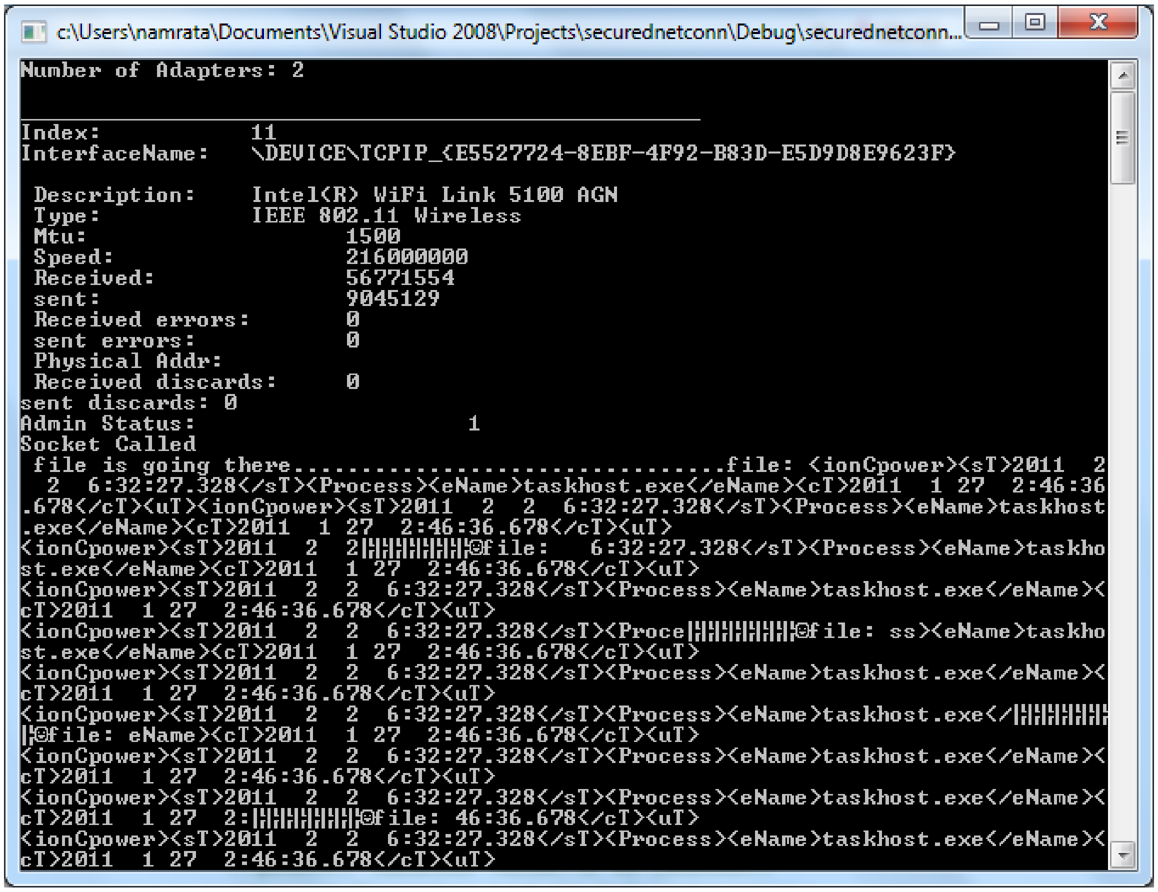
```
Completed connections          : 0
Total byte count               : 0

--- Mon Apr 18 15:49:01 2011
Current connections            : 0
Completed connections          : 0
Total byte count               : 0

--- Mon Apr 18 15:49:06 2011
Current connections            : 0
Completed connections          : 0
Total byte count               : 0

--- Mon Apr 18 15:49:11 2011
Current connections            : 0
Completed connections          : 0
Total byte count               : 0

--- Mon Apr 18 15:49:16 2011
Current connections            : 0
Completed connections          : 0
Total byte count               : 0
```

Figure 15: The server is started and socket is opened for the client to send the data

2) The time taken for the encrypting a single data block with a size of 256 bytes is less than **2 msec.**

3) The client sends the data after testing if the network is free to send the data without interrupting the user usage by checking the condition if IN and OUT defects; and IN and OUT errors are less than 2, the data packets are transferred from the client to server.

   In figure 17, the details about the active network interface are available on the client side as well as details like network interface speed, number of data bytes end and received along with IN and OUT errors and discards. It also shows the data blocks that get transferred from client to server.

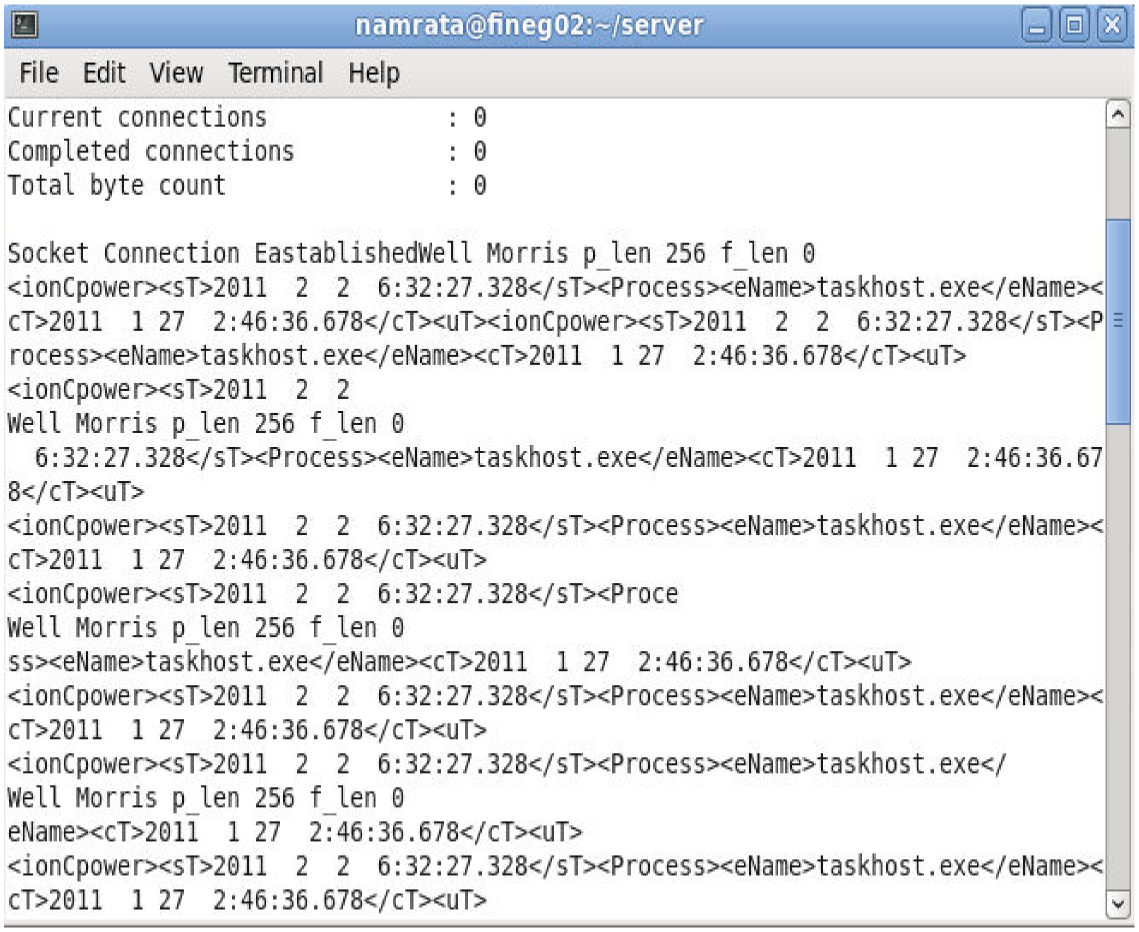


```
c:\Users\namrata\Documents\Visual Studio 2008\Projects\securednetconn\Debug\securednetconn...
Number of Adapters: 2

__________________________________________________
Index:              11
InterfaceName:      \DEVICE\TCPIP_{E5527724-8EBF-4F92-B83D-E5D9D8E9623F}

 Description:       Intel(R) WiFi Link 5100 AGN
 Type:              IEEE 802.11 Wireless
 Mtu:                      1500
 Speed:                    216000000
 Received:                 56771554
 sent:                     9045129
 Received errors:          0
 sent errors:              0
 Physical Addr:
 Received discards:        0
sent discards: 0
Admin Status:                       1
Socket Called
 file is going there.................................file: <ionCpower><sT>2011   2
  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><cT>2011  1 27  2:46:36
.678</cT><uT><ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost
.exe</eName><cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2¦¦¦¦¦¦¦¦¦¦☺file:   6:32:27.328</sT><Process><eName>taskho
st.exe</eName><cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><
cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Proce¦¦¦¦¦¦¦¦¦¦☺file: ss><eName>taskho
st.exe</eName><cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><
cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</¦¦¦¦¦¦¦¦
¦☺file: eName><cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><
cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><
cT>2011  1 27  2:¦¦¦¦¦¦¦¦¦¦☺file: 46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><
cT>2011  1 27  2:46:36.678</cT><uT>
```

Figure 16 : Client

4) The data is received on the server side and then decrypted. The decryption time on the server side for each block of 256 bytes takes less than **2 msec.**

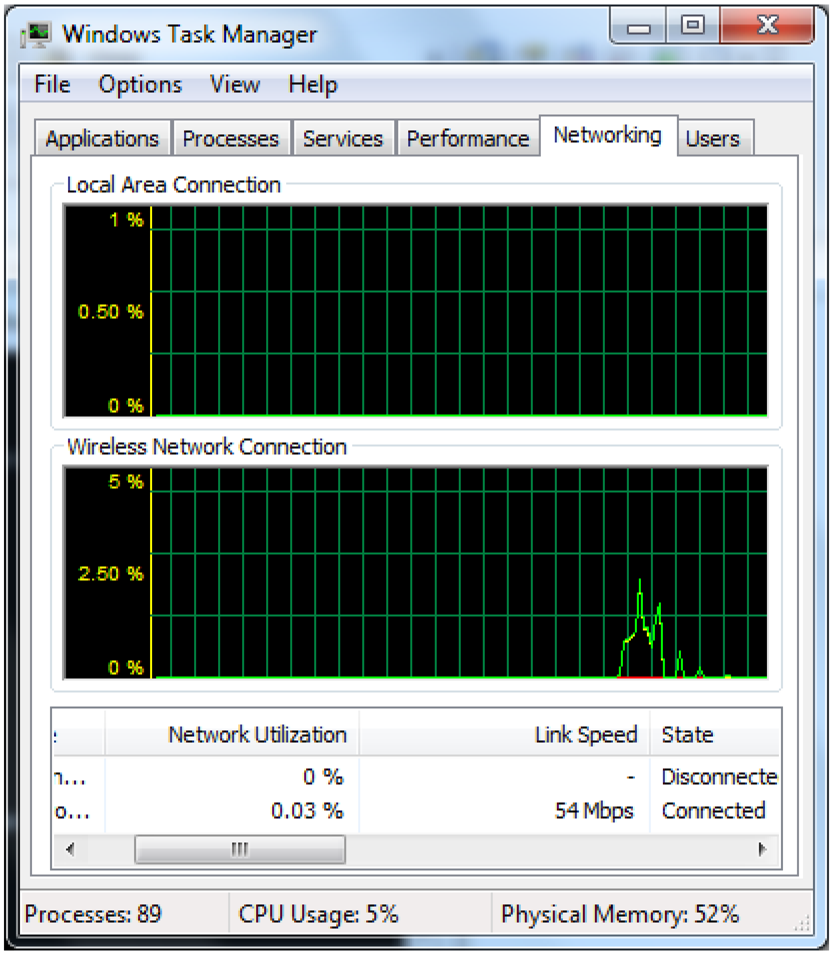
5) Figure 18 shows the data packets received on the server. These data packets are stored at the server side for multi-client architecture and dumped into MYSQL database for the further analysis purpose after performing decryption at the server side.



```
Current connections          : 0
Completed connections        : 0
Total byte count             : 0

Socket Connection EastablishedWell Morris p_len 256 f_len 0
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><
cT>2011  1 27  2:46:36.678</cT><uT><ionCpower><sT>2011  2  2  6:32:27.328</sT><P
rocess><eName>taskhost.exe</eName><cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2
Well Morris p_len 256 f_len 0
  6:32:27.328</sT><Process><eName>taskhost.exe</eName><cT>2011  1 27  2:46:36.67
8</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><
cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Proce
Well Morris p_len 256 f_len 0
ss><eName>taskhost.exe</eName><cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><
cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</
Well Morris p_len 256 f_len 0
eName><cT>2011  1 27  2:46:36.678</cT><uT>
<ionCpower><sT>2011  2  2  6:32:27.328</sT><Process><eName>taskhost.exe</eName><
cT>2011  1 27  2:46:36.678</cT><uT>
```

Figure 17: Received data

6) The network bandwidth utilization when sending the 10 packets of data through the network is around 2.0% during its peak - which is shown in figure 18.

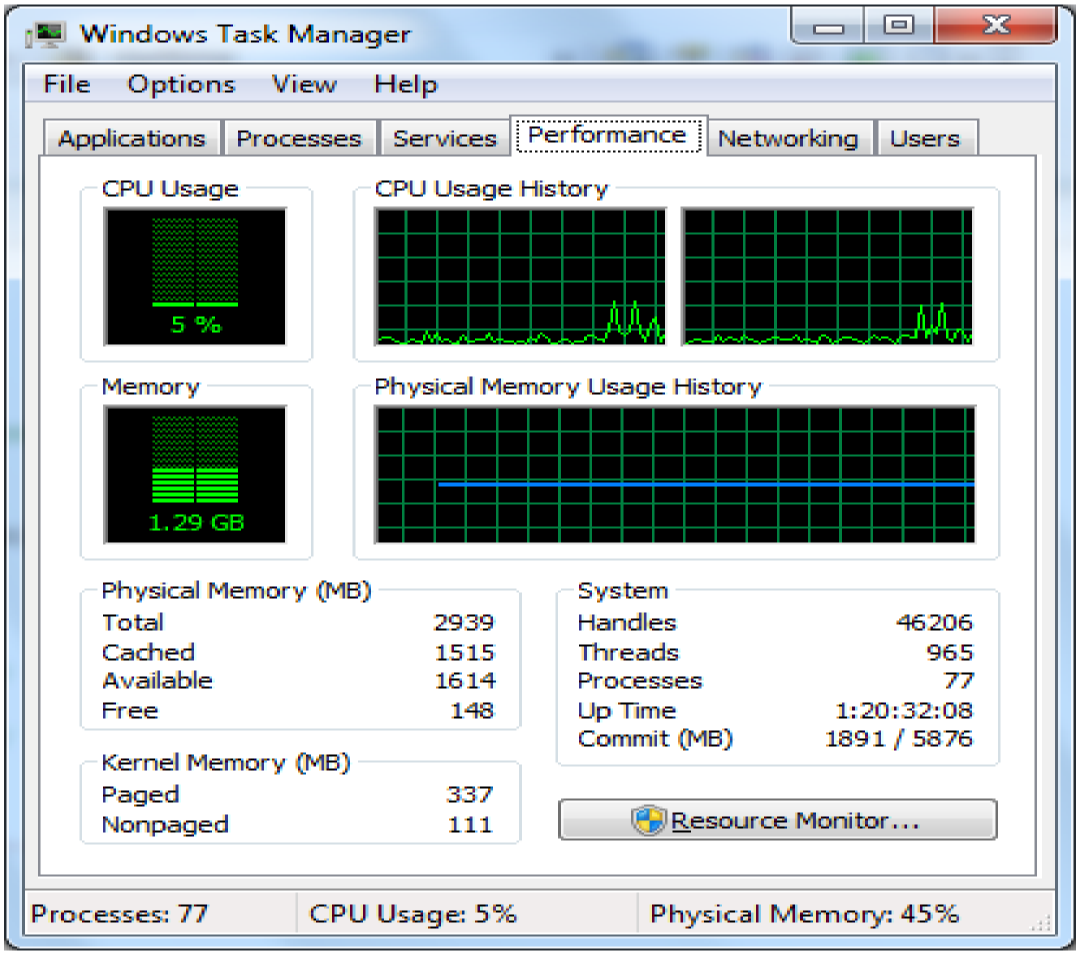

Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Local Area Connection
1 %
0.50 %
0 %
Wireless Network Connection
5 %
2.50 %
0 %

:	Network Utilization	Link Speed	State
ı...	0 %	-	Disconnecte
o...	0.03 %	54 Mbps	Connected

Processes: 89 CPU Usage: 5% Physical Memory: 52%

Figure 18: Network utilization

7) Figure 19 shows the CPU utilization while sending the data over the network is between 2-16%.



Physical Memory (MB)

Total	2939
Cached	1515
Available	1614
Free	148

Kernel Memory (MB)

Paged	337
Nonpaged	111

System

Handles	46206
Threads	965
Processes	77
Up Time	1:20:32:08
Commit (MB)	1891 / 5876

Processes: 77 CPU Usage: 5% Physical Memory: 45%

Figure 19: CPU Utilization

The feedback will be provided to each user after analyzing the data for optimizing his/her usage and reducing the energy waste.

7. Conclusion

The data collection and secure transmission by utilizing the network to its optimum is successfully accomplished for further experimental purposes. Hence, iGreen, the research project, at the intersection of technology and behavioral science, will yield insights on user behavior as well as the incorporation of user feedback as part of laptop design for energy efficiency.

8. Future Work

Enhancements are required in three different areas in this project which are explained as below:

1) **Encryption key management:** Public key cryptography algorithm, AES, is used in this project. Currently, the project works for a single client only. In future, multiple clients will access this application making it necessary to generate multiple private keys, one for each client for performing cryptography.

2) **Error recovery in control system:** There is no error recovery feature supported for the Collection and Transmission module, currently. In this model, where only single client architecture is supported, there are no chances of errors. But this might change when dealing with multiple clients. The error recovery system will be useful when such errors are generated. Thus, it is an important feature that should be supported in future.

3) **Self removal of data at the end of study period:** The data is analyzed on server side and feedback is provided to clients on how to improve the user behavior to save laptop energy power. After this study period, it is recommended to delete that data and uninstall the application as they consume unnecessary space. Thus, in future, cleaning the memory by removing this data and uninstalling the application should be implemented to maintain appropriate functionality.

9. References

1. Roth, K. W. (2007). Roth, K. W., & McKenney, K. (2007, January). *Energy consumption by Final report to the Consumer Electronics Corporation (TIAXX Reference Report D5525).* . MA: TIAX LCC: Cambridge.

2. Corportation, G. (2008, June 23). *PCs in Use Pass 1 Billion Mark.*

3. Webber, C. A. (2001). *Field surveys of office equipment operating patterns, LBNL Paper LBNL-46930* . Retrieved from http://escholarship.org/uc/item/3r7480zm

4. Roth, K.W. (2002). *Energy consumption by office and telecommunications equipment in commercial buildings: Volume I (Arthur D. Little Reference No. 72895-00).* Cambridge, MA: Arthur D. Little, Inc.

5. Kawamoto, K.K. (2001). *Electricity used by office equipment and network equipment in the U.S.: Detailed report and appendices.* Berkeley, CA: Lawrence. Berkeley, CA: Lawrence Berkeley National Laboratory: (LBNL Paper LBNL-45917).

6. Collins, R. (2007, September). *Analysis of the potential for minimum energy performance standards for computers and monitors.* Umina Beach, NSW 2257, Australia: Punchline Energy.

7. Ranganathan, P.G. (2006). *Energy-aware user interfaces and energy-adaptive displays, Computer*, 39(3), 31- 38.

8. Iyer, S. L. (2003). Energy-adaptive display system designs for future mobile environments. *The First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003) Proceedings*, (pp. 245-258).

9. *Condor High Throughput Computing.* (n.d.). Retrieved April 2011, from Idle Machine Statistics: http://www.cs.wisc.edu/condor/goodput/idle.html

10. Purchase, H. A.-K. (2008). *Information Visualization.* (Vol. Volume 4950/2008).

11. Darby. *OP cit.*

12. Fischer, C. (June 4-5, 2007.). *Consumer feedback: A helpful tool for stimulating electricity conservation. Cases in Sustainable Consumption and Production: Workshop of the Sustainable Consumption Research Exchange (SCORE!) Network.* Paris, France,.

13. Fischer, C. (4-9 June 2007). *Influencing electricity consumption via consumer feedback. A review of experience.* ECEEE 2007 Summer Study.

14. Fogg, B.J. (2002). *Persuasive technology: Using computers to change what we thing and do.* San Francisco: Morgan Kaufmann.

15. van Houwelingen, J.H. (1989). *The effect of goal-setting and daily electronic feedback on in-home energy use.*. The Journal of Consumer Research.

16. Midden, M. &. *Op cit.*

17. *AES encryption and decryption.* (2011). Retrieved April 2011, from Developer Zone: http://http.developer.nvidia.com/GPUGems3/gpugems3_ch36.html

18. *bandwidth*. (2011). Retrieved April 2011, from About.com: http://compnetworking.about.com/od/speedtests/g/bldef_bandwidth.htm

19. Morris Jones, B.W. (2010). *Pilot Study of College student Laptop Usage for Energy Efficiency.* San Jose, California.

20. *Optimal Transfer Algorithm*. (2011). Retrieved April 2011, from CommServer: http://www.commsvr.com/Howitworks/Technologie/Optimization/OTA.aspx

21. *Quality of service*. (2011). Retrieved April 2011, from Wikipedia: http://en.wikipedia.org/wiki/Quality_of_service

22. *Resource Availability Troubleshooting Getting Started Guide*. (2011). Retrieved April 2011, from MSDN website: http://technet.microsoft.com/en-us/library/dd883276(v=ws.10).aspx

23. Solworth, M. R. (2004, September 24). *Socket Programming in C/C++*. Retrieved April 2011, from http://www.rites.uic.edu/~solworth/sockets.pdf

24. *Windows Task Manager*. (2011). Retrieved April 2011, from Wikipedia: http://en.wikipedia.org/wiki/Windows_Task_Manager

25. Caruso, J. B. (2009). *The ECAR Study of undergraduate students and information technology 2009.* Educause. (Mahesri)

26. Mahesri, A. &. *Power consumption breakdown on a modern laptop In B. Falsafi & T. N. Vijaykumar (Eds.), Power-aware computer systems.* Berlin:: Springer-Verlag.

27. Bardini, T. &. (1995). *The social construction of the personal computer user* . Journal of Communication.

28. Jain, D. A. (January 2000). *Statistical Pattern Recognition: A Review.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 1,.

29. Webb, A. (2002). *Statistical Pattern Recognition,.* John Wiley & Sons, 2002, 2nd Edition, .

30. *Process Status API*. (2011). Retrieved April 2011, from MSDN Library: http://msdn.microsoft.com/en-us/library/ms684884(v=vs.85).aspx

31. *IP Helper*. (2011). Retrieved April 2011, from MSDN Library: http://msdn.microsoft.com/en-us/library/aa366073(v=vs.85).aspx

32. *MIB Structures*. (2011). Retrieved April 2011, from MSDN Library: http://msdn.microsoft.com/en-us/library/aa366889(v=VS.85).aspx

33. *Condor High Throughput Computing. (n. d.).Idle Machine Statistics* http://www.cs.wisc.edu/condor/goodput/idle.html

34. *Snowball sampling*. (2011). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Snowball_sampling