9/1 (2011), 77–90
tmcs@math.klte.hu
http://tmcs.math.klte.hu

**Teaching
Mathematics and
Computer Science**

Facilitating class attendance to improve student achievements

Ye Sun and Weifeng Chen

Abstract. Many studies have revealed that attendance is strongly associated with students' achievements, and have proposed different strategies to improve students' attendance. However, there are few studies investigating how to efficiently take students' attendance – the key component to improve students' attendance. Taking attendance manually is inefficient since it will consume part of the limited class time. This paper describes the design and the implementation of an online attendance system that is currently used in classes at West Virginia University and California University of Pennsylvania. Examples of the system are provided online. Implementation codes of the system are shared, which can be used to teach computer science courses such as Web Programming or Client-Server Script Languages.

Key words and phrases: online attendance, educational technology.

ZDM Subject Classification: U50, N80.

1. Introduction

The importance of attendance to students' performance is well understood. Many studies have revealed that attendance is strongly associated with students' achievements [1, 2, 5, 6, 7, 8, 10, 12], since the most invaluable time for students' achievements is the time spent in the classroom, compared to other time such as time spent in discussion sections or time spent studying outside of class preparing for the class session itself [11].

Consequently, strategies to improving students' attendance have been proposed [9]. However, there are few studies investigating how to take students' attendance, which is the key component to improve students' attendance.

Manually taking attendance is a time-consuming process, especially for a big class in which the instructor may not recognize every student. In a 50-minute class, it may take 5 minutes to take students' attendance, which is very inefficient, considering that the most valuable and important time commitment in a course was the time actually spent in the classroom [11]. In this paper, we describe a simple online attendance system that is currently used in classes at California University of Pennsylvania (CalU) and West Virginia University (WVU). Design and implementation of the system are presented.

With the aid of computer technologies, most of the classrooms at CalU and WVU are equipped with computers with Internet connections. The online attendance system allows students to take their attendance from the computer they are using with a simple click. Although CalU and WVU pay money to use some online education systems such as Blackboard, E-College or Desire2Learn, none of them has the function for students to take attendance. The one described in this paper is implemented using simple script languages. All of the codes are shared online and can be used as samples to teach programming courses in computer science.

The rest of the paper is organized as follows. In Section 2, we present the design of the system. Different requirements of the system are proposed. Section 3 describes the implementation of the system that satisfies those requirements. A couple of related issues are discussed in Section 4. We finally conclude our paper in Section 5.

2. Design

In this section we describe the requirements of an online automatic attendance system, based on which we will present the modules of the system.

2.1. System Requirements

Normally, in a classroom equipped with computers, each student will use a separate computer. Considering such a scenario, we define the following attendance policies.

1. An attendance can only be taken during the class time. Students should not be able to take attendance before or after the class.
2. Each student can take at most one attendance in one class.
3. Attendances can only be taken within the classroom. In other words, only the computers physically located in the classroom can be used to take attendance.

The main purpose of this online attendance system is to save the instructor's time so that the instructor can efficiently spend the whole class time in teaching. It also needs to facilitate the instructor's managing students' attendance in some special situations, for example, students absence due to sickness; students forgot to take attendance but realize it immediately when the class is over; some computers are broken and two students share a single computer. When these situations happen, the instructor may also want to manage the attendance records.

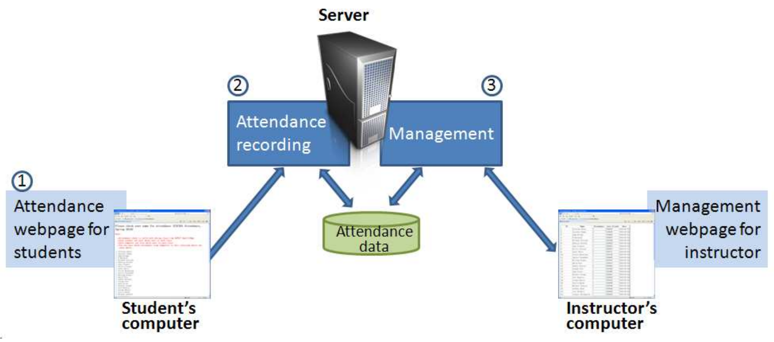


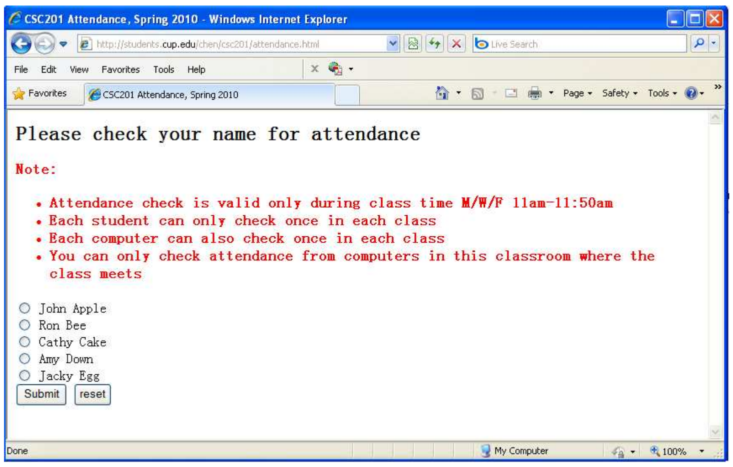
Figure 1. Models of the system

2.2. Model

The requirements mentioned in the previous subsection will be achieved by the attendance models described in this subsection. This attendance system consists of three major components (Figure 1). Attendance webpage is the webpage students will use to take their attendances. Every student's name is listed on the webpage and at most one student can be selected. After a student selects his/her own name and submits the attendance, the information is sent to the second component, a common gateway interface (CGI) program written by Perl. The CGI program running on the web server checks whether an attendance is valid based on the attendance policies described previously. If the attendance is valid, the CGI updates the attendance records; otherwise, it sends an error message back to the student's web browser. The third component is the management component. Basically, this component allows the instructor to modify students' attendance in the special scenarios described above. It also displays some statistics data of the attendance, e.g., the maximum, the minimum and the average of students' attendances.

3. Implementation

This section describes the implementation of the three components shown in Figure 1. Codes used to implement the system will be presented.


CSC201 Attendance, Spring 2010 - Windows Internet Explorer
http://students.cup.edu/chen/csc201/attendance.html
Live Search
File Edit View Favorites Tools Help
Favorites
CSC201 Attendance, Spring 2010
Page Safety Tools
Done
My Computer
100%

Please check your name for attendance

Note:

- Attendance check is valid only during class time M/W/F 11am-11:50am
- Each student can only check once in each class
- Each computer can also check once in each class
- You can only check attendance from computers in this classroom where the class meets

○ John Apple
○ Ron Bee
○ Cathy Cake
○ Amy Down
○ Jacky Egg

Submit reset

Figure 2. Attendance webpage

3.1. Component one: Attendance webpage

The first component – attendance webpage (Figure 2)– is quite simple. Basically, it lists all the students' names and specifies the CGI program (the second component of the system) to which the attendance information will be sent. The webpage uses an HTML form with a submit button and a reset button. All of the student names are listed using the input check box in the form (lines 11 to 15

```
1. <body>
2. <form action = “../cgi/attd/attendance.pl” method = “post”>
3. <h2> Please check your name for attendance </h2>
4. <h3><font color=red> Note:
5. <ul>
6.     <li> Attendance is valid only during class time M/W/F 11am-11:59am </li>
7.     <li> Each student can only check once in each class</li>
8.     <li> Each computer can also check once in each class </li>
9.     <li> You can only check attendance from computers in the classroom </li>
10.</ul></font></h3>
11. <input type = ”radio” name = ”student” value=”1_John Apple”/> John Apple <br />
12. <input type = ”radio” name = ”student” value=”2_Ron Bee”/> Ron Bee <br />
13. <input type = ”radio” name = ”student” value=”3_Cathy Cake”/> Cathy Cake <br />
14. <input type = ”radio” name = ”student” value=”4_Amy Down”/> Amy Down <br />
15. <input type = ”radio” name = ”student” value=”5_Jacky Egg”/> Jacky Egg <br />
16. <input type = ”submit” value = ”Submit” />
17. <input type=”reset” value=”reset” />
18. </form>
19.</body>
```

Figure 3. Source code of the attendance webpage.

in Figure 3) [1], so that at most one student name can be checked. Each student input box is also associated with a value, which was used by the CGI program to recognize the student. The action attribute of the form indicates the location of the CGI program where the data (i.e., the student name that is checked) will be sent. In this particular example, line 2 in Figure 3 indicates that the CGI program is named *attendance.pl*, written by Perl and located in directory “../cgi/attd/”.

Readers can visit `http://students.calu.edu/chen/CSC201/stuattend.html` to see the real webpage. To view the source codes, one can right click on the webpage and select the “View Source” option from the pop-up menu.

Note that the value of each radio consists of a number index and the name of the student. This is to forward the name of the student to the CGI program on the server, so that the CGI program can record the attendance of the correct student. Details will be described next.

3.2. Component two: CGI programs processing student attendance

When the CGI program receives the radio value sent from the attendance webpage, it will obtain the student index and student name using the following two lines of codes.

[1]Students names here are faked due to privacy concern.

```
my(\$student) = (param("student"));
chomp((\$studentindex, \$stdname) = split(/_/, \$student));
```

The obtained $stdname will be used to compare with the names stored on the server to make sure that an attendance is recorded for the correct student, after the three conditions described in Section 2.1 are validated. Figure 4 shows the dataflow of the CGI program, which contains three functions to check the three conditions. These functions will be briefly described in the following. The complete source codes of the CGI program can be viewed at `http://students.calu.edu/chen/csc201/attendance.pl.txt`.

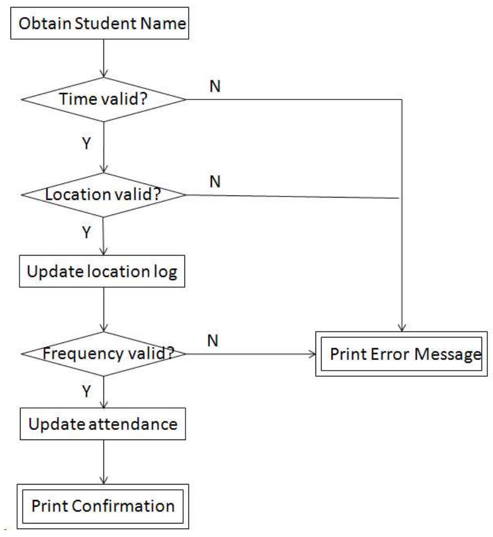


Figure 4. Data flow of the CGI program

Time requirement

The time requirement ensures that a student can only take attendance during the class time. This requirement is validated within function *validtime*() shown in Figure 5. After the current time is obtained in Line 2, the IF statement in Lines 4-7 checks whether the time falls in the slot of Monday/Wednesday/Friday 11am-11:59am. If it does, the current time in seconds, produced by function *timegm*() in Line 3, will be returned, which will be recorded in the attendance data file.

```
sub validtime {
   ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
   $currentsec = timegm ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) ;
   if (( (($wday == 1)||($wday == 3)||($wday == 5)) && ($hour==11))) {
          return $currentsec; }
   else {
          return 0; }
}
```

Figure 5. Function *validtime*() in the CGI program that verifies the time requirement.

Otherwise, an error message is sent back to the student and the CGI program stops.

Location requirement

If the time is valid, the CGI program continues to check whether the location requirement is met. An attendance meets the requirement if

1. it comes from a computer located in the classroom; and
2. no attendance have been previously submitted from the computer in the same class.

The first condition can be checked based on the IP addresses of the computers [2]. The IP addresses of the computers within the classroom were obtained from the IT service department at CalU. In fact, these IP addresses are consecutive since they are assigned as a block to the computers within a same room. Specifically, the IP addresses of the computers in the classroom studied in this paper are from 158.83.180.101 to 158.83.180.150. Line 2 in Figure 6 verifies the IP address of the computer from which an attendance request was submitted.

It is a little complicated to verify the second condition. To check whether an IP address has submitted an attendance request, we maintain a log file *addresslog.txt* to record an IP address and the last moment when a valid attendance request was submitted from this IP address. Table 1 shows sample records in the log file *addresslog.txt.* The last record means that, at time 1272272539 (GMT

[2]Here, we assume that these IP addresses are global IP addresses, which are the case for most of the universities in U.S.. Section 4 will discuss the Network Address Translator (NAT) case.

```
sub validsource {
1.    chomp(( $byteone, $bytetwo, $bytethree, $bytefour) = split(/\./, $_[0]));
2.    if ( ($byteone==158)&&($bytetwo==83)&&($bytethree==180)
                  &&($bytefour> 100)&&($bytefour< 150)) {
3.        open(IPDAT, “+<addresslog.txt”) or error(0);
4.        flock(IPDAT, LOCK_EX) or error(1);
5.        $count=0;
6.        foreach $line (<IPDAT>) {
7.            chomp(($address, $lasttime) = split(/:/, $line));
8.            $addresslog{$address}=$lasttime;
9.            if ($address eq $_[0]) {
10.               $count=1;
11.               if ( ($_[1]-$lasttime)<3000 ) {
12.                   close(IPDAT);
13.                   error(3);
14.               } else
15.                   $addresslog{$address}=$_[1];
16.           }
17.       }
18.       if ($count == 0)
19.           $addresslog{$_[0]}=$_[1];
20.       seek(IPDAT, 0, 0) or error(2);
21.       foreach $address (keys % addresslog)
22.           print IPDAT “$address:$addresslog {$address}\n”;
23.       close(IPDAT);
24.       return 1;
25.   }
26.   else
27.       return 0;
}
```

Figure 6. Function *validsource*() in the CGI program to verify the location requirement.

format in seconds), a valid attendance request was submitted from IP address 158.83.180.96.

Based on this log file, the second condition for the location requirement is verified from Line 3 to Line 23 in Figure 6. Specifically, if an attendance request is sent from an IP address that has been recorded in the log file, the current time, which is returned by Function *validtime*() described in Figure 5, is compared to the IP address’s last moment recorded in log file *addresslog.txt.* If no attendance request has been previously submitted from the IP address in the current class,

Table 1. Log file *addresslog.txt*

158.83.180.91:1272273237
158.83.180.47:1272013734
...
158.83.180.96:1272272539

the time difference should be greater than 3000 seconds, i.e., 50 minutes, one class period. In this case, the current time will be updated as the new moment for the IP address in the log file (Line 15). If the time difference is less than 3000 seconds, an attendance has been previously submitted from the IP address in the current class. An error message will then be displayed (Line 13). Lines 18-19 deal with the scenario when the IP address has not been recorded in the log file. This happens when students take attendances the first time in the first class.

Frequency requirement

After an attendance request passes the time requirement and the location requirement, it needs to go through the frequency requirement, i.e., each student can take at most one attendance in one class. Similar to the location requirement described above, we maintain another log file to record students and their last moments of valid attendance, as demonstrated in Table 2. Each record in the file contains four fields, separated by a column symbol “:” – an index, student’s name, student’s attendance, and the most recent time (in GMT format seconds) when the student takes a valid attendance. For example, the first record in Table 2 indicates that student “John Apple” has taken three attendance, and the last time he took the attendance is 1263979605 seconds in the GMT format. The last record indicates that student “Jacky Egg” has not taken any attendance; his most recent attendance time is a -1. Consequently, the frequency requirement of a student can be verified by comparing the current time to the student’s most recent attendance time in file *attendance.txt*. If the time difference is greater than 3000 seconds, it is a valid attendance; otherwise, an error message is printed.

Table 2. File *attendance.txt* recording students's attendances

```
1:John Apple:3:1263979605
2:Ron Bee:1:1263978436
3:Cathy Cake:1:1263979635
4:Amy Down:1:1263979562
5:Jacky Egg:0:-1
...
```

3.3. Component three: Attendance management

So far we have described the first two components of this online attendance system. Due to the reasons mentioned in Section 2.1, our online attendance system also includes the third component – Management module – for instructors to effectively manage students' attendance data. For example, an instructor can see the number of a student's successful attendance and the time of his/her last attendance. At California University of PA, faculty members are asked to verify the roster in the beginning of every semester. With this attendance system, it is very easy for an instructor to identify whether a student is always attending, has stopped attending or never attends. An instructor can also update attendance data in the case when some students cannot come to class because of sickness or two students are sharing one computer. In the latter case, due to the location requirement described in Section 3.2, the second student cannot take the attendance. An instructor can also display statistics information about students' attendance in the class, such as average attendance, maximum attendance and minimum attendance, etc.

It is obvious that this management module needs password protection so that only authorized persons can manage the attendance. In our implementation, the management module consists of three files: *login.htm, attdupdate.php* and *attddisplay.php.* Figure 7 shows the three components and the data flow among them. *Login.htm* asks for a username and password to update the attendance information. The input username/password are sent to *attdupdate.php.* We use the PHP program just in order to demonstrate that this simple attendance system can be implemented using different programming languages. The *attdupdate.php* verifies the username and password. If they are correct, it will display students' attendance information in a changeable way (e.g., in a text input field) so that the instructor can change the attendance information. Once information is updated

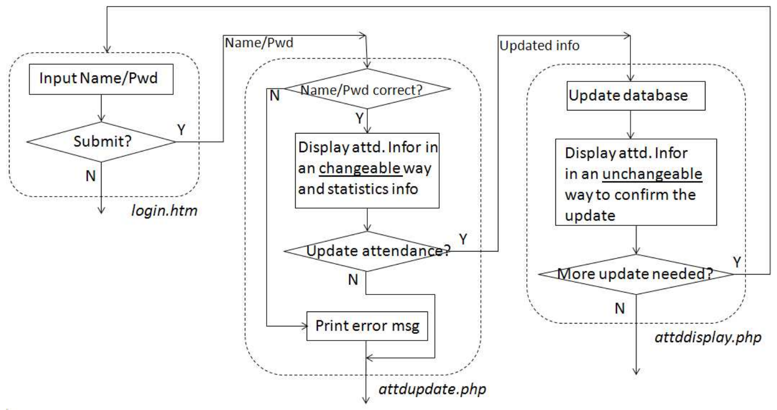


Figure 7. Program components and data flow of the management module

and confirmed, it will be sent to the *attddisplay.php*, which updates the attendance information in attendance.txt and displays the updated attendance in an unchangeale format to confirm that the new attendance data has been recoded. It also brings to the *login.htm* if more updates are needed.

3.4. Summary

To summarize, we have described the three components of the online attendance system. A real system is available online at `http://students.calu.edu/chen/csc201/attendance.htm`. To test the management module, a username “attadmin” and password “csc201” are needed.

4. Discussion

In this section, we will discuss additional issues related to the implementation of the attendance system.

4.1. Hosting Server

Currently the attendance system used at CalU is hosted at the CalU website (`http://students.calu.edu/`), which supports different scripting languages including Perl and PHP. If it is impossible to host the system on a university website, there are lots of free web hosting on the Internet that supports various scripting languages [4].

4.2. Obtaining IP addresses

In Section 3.2 we described that the location requirement of the attendance system is achieved based on the IP addresses of the computers physically located in the classroom. At CalU, we obtained the IP addresses of the computers in classrooms through our IT service department, which maintains the computers and networks of the university. If there is no such department at a university that can provide the IP addresses of the computers, the program can be easily modified to collect IP addresses automatically. In particular, the CGI program that verifies students’ attendance will remove the location requirement check (Line 2 in Figure 6) and use statement $addr = $ENV{’REMOTE_ADDR’} to obtain the IP addresses of the computer from which an attendance request is sent. Then in the first day of a class, the instructor can ask students to submit their first attendance, which will sent the IP addresses of the computers in the classroom to the CGI program. After that, the CGI will have the list of valid IP addresses and the location requirement can be enforced.

Note that this approach is good enough even though dynamic host configuration protocols (DHCP) are used, due to the following reasons. Desktop computers in most classrooms connect to the Internet using wired connections. Their IP addresses are always the same under DHCP. Even for wireless connections under DHCP, in most of the DHCP implementations (e.g., the automatic allocation approach), a client (e.g., a laptop) will be assigned a previously-allocated IP address by the DHCP server. The client will be allocated a different IP address only when the previously allocated address has been allocated to other computers that are currently active in the network, which only happens when the DHCP server has depleted all the available IP addresses in the address pool, an extremely rare case in real life.

4.3. IP addresses behind Network Address Translator (NAT)

So far, we assume that IP addresses of computers in the classroom are global IP addresses (i.e., each computer has a distinct IP address), which are the case for most of the universities in U.S.. It is known that IP addresses of different computers behind a Network Address Translator (NAT) box appear the same, which is the IP address of the NAT box, to a host outside the NAT box. This seems to violate the location requirement. However, this system is mainly designed to be used to take attendance of students taking classes in a classroom at a university. This means that the server and the computers in the classroom will be in the same network. If the university uses a NAT box, both the server and the computers are behind the NAT box. In this case, the server is still able to validate the location requirement based on the internal IP addresses of the computers, which are distinct.

4.4. Multiple courses to use the attendance system

So far we have focused on only one course. If multiple courses want to use our proposed attendance system, each course will have a separate attendance system. Basically, each course will have separate software components shown in Figure 1 due to the following considerations: 1) Each course will have different rosters; 2) Instructors may be different so that the management login information may be different; and 3) Class periods may be different. In this case, the three requirements presented in Section 3.2 are still valid. Specifically, if a student are taking two consecutive courses being held in a same classroom, the student is able to take his/her attendance for both courses one after the other, via two different webpages.

4.5. CAPTCHA to prevent machine attendance

It is interesting to consider the case that a student writes a robot (a small computer program) that runs on a computer in the classroom and takes the attendance automatically without the student sitting in the classroom. Due to the location requirement (Section 3.2), the automatic attendance taken by the robot will also result the student who physically sit at the computer not able to take his/her attendance. A CAPTCHA, "a type of challenge-response test used in computing as an attempt to ensure that the response is not generated by a computer" [3], is the straightforward and effective approach to address this problem. Currently we have not implemented CAPTCHA in the system since students do not have any privileges to install any program on campus computers.

5. Conclusion

In this paper we have described the design and implementation of an online attendance system that allows students to take attendance on Internet-connected computers with a simple click. The implemented system will save instructor's time so that the whole class time could be used efficiently to improve students' achievements. URLs of the system currently used at California University of PA have been provided. Codes of the system are shared, which could be used as examples to teach computer courses such as Web Programming or Client-Server Script Languages.

References

[1] N. Ali, K. Jusof, S. Ali, N. Mokhtar and A. Salamat, The factors influencing students' performance at University Teknologi MaraKedah, Malaysia, *Management Science and Engineering* **3**, no. 4 (2009), 81–90.

[2] L. W. Buckalew and J. D. Daly, Relationship of initial class attendance and seating location to academic performance in psychology classes, *Bulletin of the Psychonomic Society* **24**, no. 1 (1986), 63–64.

[3] CAPTCHA, Wikipedia, `http://en.wikipedia.org/wiki/CAPTCHA`.

[4] Free Web Hosting, `http://www.free-webhosts.com/webhosting-01.php`.

[5] R. C. Johnson, As studies stress link to scores, districts get tough on attendance, *Education Week* **20**, no. 7 (2000), I, 10.

[6] H. C. Jones, Interaction of absences and grades in a college course, *Journal of Psychology* **116** (1984), 133–136.

[7] R. E. Ledman, Improving student attendance: Does it improve student learning?, *Academic Exchange Quarterly*, Retrieved August 09, 2010 from the World Wide Web, `http://findarticles.com/p/articles/mi_hb3325/is_1_6/ai_n28914618/`.

[8] K. H. Park and P. M. Kerr, Determinants of Academic Performance: A Multinomial Logit Approach, *The Journal of Economic Education* (Spring 1990), 101–111.

[9] D. B. Reeves, Improving Student Attendance, *Educational Leadership* (May 2008), 90–91.

[10] D. Romer, Do Students Go to Class? Should They?, *The Journal of Economic Perspectives* (Summer 1993), 167–174.

[11] R. M. Schmidt, Who Maximizes What? A Study in Student Time Allocation, *American Economic Review* (May 1983), 23–28.

[12] D. R. Street, Non-compulsory attendance: Can state supported universities afford this luxury?, *Journal of College Student Personnel* (March 1975), 124–127.

YE SUN
DEPARTMENT OF CURRICULUM, INSTRUCTION & LITERACY
WEST VIRGINIA UNIVERSITY, MORGANTOWN, WV 26505, USA

E-mail: yesun@mail.wvu.edu

WEIFENG CHEN
DEPARTMENT OF MATHEMATICS, COMPUTER SCIENCE AND INFORMATION SYSTEMS
CALIFORNIA UNIVERSITY OF PENNSYLVANIA, CALIFORNIA, PA 15419, USA

E-mail: chen@calu.edu

(Received August, 2010)