Student Elections Online

Dickinson College uses technology to get out the vote

by **Paul Dempsey**

Getting college students to participate in elections is a continual challenge for student governments. Voting rates of 30 percent are considered good and those in the single digits are not unheard of. This student apathy may only be a reflection of society at large. Presidential elections attract half the eligible voters, while off-year congressional races bring out little more than a third.[1]

In an attempt to combat this apathy and increase voter participation, student government often turns to technology for answers. Information technology staffs are getting requests to put student elections on the Web.[2] Given the various projects that compete for our attention, how should these requests be handled?

At Dickinson College in Carlisle, Pa., we conducted our first Web-based election in April 2000. The "polls" were open for 48 hours and more than 51 percent of the students voted. Election results were available immediately after polls closed. We considered the project important as a way to increase participation in elections as well as showcase the college's investment in technology.

Pros and Cons

In the past, student government elections involved paper ballots at a single voting location, which controlled who could vote. The system was both inconvenient and susceptible to error. As technology became more prominent on college campuses, student leaders tried to find ways to automate the election process. Various options included standalone microcomputer voting, opscan ballots, telephone balloting using an interactive voice response (IVR) system, and e-mail voting.[3] The tremendous growth of the World Wide Web in recent years has made it the latest technology to be enlisted.

In devising a security system, the challenge is striking a balance between security and convenience.

The Web provides a number of clear advantages, including a central server to provide control and widespread access from almost any computer with an Internet connection. Disadvantages include potential security violations and the possibility of disenfranchising voters who are unable or unwilling to use the technology. Web voting success on any particular campus depends in part on the campus environment. If you have a large number of students without access to computers or a history of hotly contested elections, then Web voting may not work.

Dickinson College is a private liberal arts college with about 2,000 students. All residence halls are networked and a substantial number of students have computers in their rooms. In addition, networked computers are available in a number of labs as well as the library and the student union. We felt that using the Web would not exclude any students from the process.

Planning for Online Elections

In cities with a history of political corruption, the motto on Election Day was "vote early and vote often." Obviously, any Web-based election has to provide basic security so that only eligible students can vote and each voter can cast only a single ballot. In devising a security system, the challenge is striking a balance between security and convenience.

We decided to use a student's e-mail address as the unique ID for the election. A four-digit PIN was generated randomly for each student and sent via e-mail. The assumption was that only

the students had access to their individual e-mail accounts. Students already knew their e-mail address and would just need to remember the PIN to vote.

Dickinson's Web server is configured with an internal section that is only available to computers on the campus network (including residence halls). After weighing the security and convenience issues, we decided to run the election on the internal section. Since Dickinson is a residential college with a small number of off-campus students, the need for external access would be small and was outweighed by the additional protection this set-up would provide against unauthorized use.

Figure 1: **Sample Sophomore Ballot**

Preparing the Ballots

Next we had to create the Web pages and programs that would be used in the election. Our environment includes a Netscape Enterprise Server running on a Unix machine, with perl as the primary programming language for the CGI (common gateway interface) applications used to process Web forms. One option was to "hard-code" the HTML form that would be used for the ballot and then create a custom perl program to process it. While this approach would be fairly straightforward, it was the least flexible.

Another possibility was to find an existing ballot program and adapt it to our needs. There are many collections of perl CGI scripts available through the Web, but a quick search for voting programs was not helpful. Most of the results were for online polls such as "who's your favorite actor/band/shortstop." These programs often allowed one vote per computer (based on each computer's unique IP address). Although this approach can prevent one person from influencing a poll, it would not work for us since many students might vote from the same computer in a lab.

We designed our own voting program that could be reused for different elections. This program relies on a data file to generate the ballot, and then uses the same file to tabulate results. For each new election, we just change the data file. This approach also allows us to generate customized ballots. Freshmen, for example, get a ballot enabling them to vote for freshman senators but not for senators from other classes.

Exhibit 1 - Student Data File

```
HEAD | Senate Officers
ALL | Senate President | 1 | Richard Russo | Jane Smiley
ALL | Senate Vice President | 1 | Jon Hassler | Carolyn Chute | Richard Price
HEAD | All College Committees
ALL | Planning and Budget | 2 | Tim O'Brien | John Updike | Anne Tyler
HEAD | Class Representatives for Student Affairs
FR | Sophomore Rep | 2 | John Sayles | T.C. Boyle | William Boyd
SO | Junior Rep | 2 | Margaret Atwood | Larry McMurtry | Don DeLillo
```

Specific Techniques

To get the list of eligible voters, the information technology division extracted data from our student information system using Datatel's Colleague software. A simple tab-delimited ASCII (plain text) data file was prepared with each student's e-mail address, PIN, class standing, first name, and last name. The PIN was a four-digit number randomly

generated by Colleague.

We designed a format for the ballot data file that would display headings and present the candidates for each office. The entry for each office included the name of the office, category of students who could vote for the office, maximum number of candidates to vote for, and candidates' names.

Exhibit 1 shows a sample ballot file. This is an ASCII file with fields delimited by the "|" character. Lines with "HEAD" as the first field indicate that the second field should be displayed as a heading; otherwise, the line is an entry for one of the offices. These entries begin with a field that shows which students may vote for that office. "ALL" indicates all students, "FR" indicates freshmen only, and so on.

The second field indicates the name of the office, such as "Senate President." The third field indicates the number of candidates the voter may select. For Senate President, for example, the voter may select one candidate, but for Planning and Budget Committee a voter may select up to two candidates.

The ballot data file performed a number of functions. It generated ballots for individual voters and presented them with the appropriate offices based on class standing. The ballot also used different HTML form elements for different offices. When only one candidate was to be selected, we offered a radio button, which only allows one choice in a category. For offices with two or more candidates, we offered a checkbox. Figure 1 shows how a sophomore ballot looked.

The data file was used again after a ballot was submitted to validate the votes, making sure a voter did not choose more than the maximum number of candidates. If too many were selected, an error message was displayed and the votes were not recorded. The ballot data were also used to tally the results when voting ended.

HTML Forms and CGI Programs

To allow students to vote, we created an HTML form for them to log in, a CGI program to process this form and generate the ballot, and a second CGI program to record the vote. Figure 2 shows the initial screen seen by the voters.

We anticipated that some students would lose their PINs prior to the election, so we added a PIN-finding feature to the login page. Students who forgot their PINs could use the form on the lower half of the page to enter their unique ID and have their PIN sent via e-mail. This turned out to be an important feature, since more than 30 percent of students who voted requested their PINs this way.

Some students would be voting from public computers in labs, so we had to consider security at this point in the process. Because not all browsers behave the same way, we tried a few different tactics. The login screen was designed to expire immediately so individual student data would not remain in the browser's cache. We also inserted a small JavaScript program in the page that would run when the "Get Ballot" button was clicked to submit the form. This program would erase the PIN from the text entry box and place it in a hidden field. The login screen was also designed to launch a new browser window for the students to vote. This would allow them to close their ballot window when they were done without exiting the browser.

The ID and PIN from the login page would then be sent to the first CGI program, which would follow these steps:

- Verify ID. If invalid, give error message.
- Verify PIN. If invalid, give error message.
- Check to see if this student already voted. If yes, give error message.
- If no errors, generate ballot based on student's class standing. Place an encrypted version of the student's ID in the form as a hidden field.

The student would then make his or her selections from the ballot.

Figure 2: **Election Logon page**

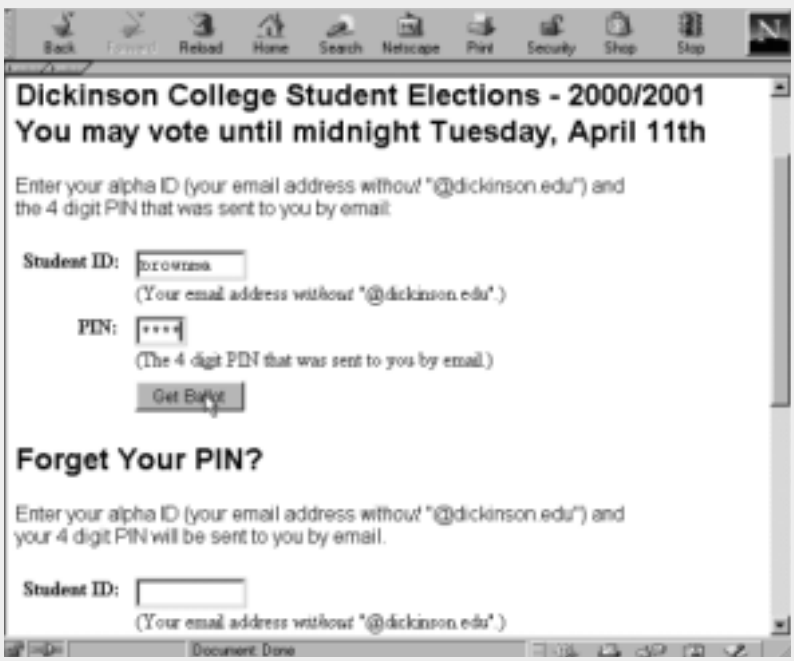
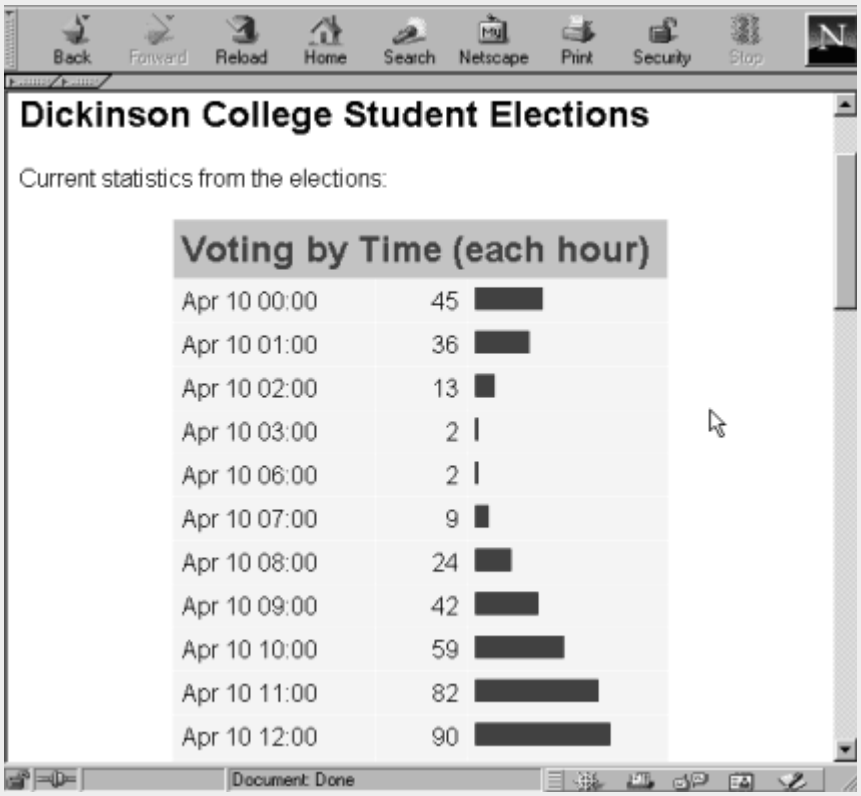


Figure 3: **Voting Statistics**



When the student submitted the ballot, the second CGI program would process it with these steps:

- Check the encrypted ID to see if valid. If not, give error message.
- Make sure student hasn't already voted. If yes, give error message.
- Check the votes against the ballot data file. If too many candidates selected for any office, give error message.
- If no errors, record votes in data file and record time voted and computer IP number in log file.

It was important for this program to check again that the student had not already voted. Otherwise, it would be possible to bypass the initial login and submit multiple votes.

Conducting the Election

Once all the forms and programs were created and tested, we were ready to hold the election. The IT staff used a feature of Colleague to generate e-mail messages to the students notifying them about online elections and providing their PINs. The message was sent the week before the election, and included a link to the online ballot. The Student Senate publicized online voting with posters distributed around campus, and the college home page promoted the election and featured a link.

Because there were no physical election locations to be staffed, we planned to begin voting at 12:01 a.m. on Monday and continue for 48 hours. A job was scheduled on the Web server to activate the login screen and programs at the appropriate time. To end the elections, a similar job was scheduled to remove the login screen and disable the programs.

Since Web-based voting was new at Dickinson, the student election committee was concerned about participation. We created a program that allowed them to monitor voting statistics while the election was taking place. This program (see Figure 3) generated statistics on the number of students from each class voting each hour.

The final issue to be addressed was the vote tallying. A program to count the ballots was executed automatically when the polls closed, and the results were posted on the Web site within seconds. The election process that once took hours of tedious (and error-prone) work was done by the computer, much to the relief of the election committee.

Post-Election Wrap

Web-based voting at Dickinson College was tremendously successful—more than 51 percent of students participated. The Student Senate accepted the results and plans to use the process again. IT staff resources were important in making the system work. We spent a few hours of programming time to extract data from the student information system and generate the e-mail messages to contact the students. The student election committee created the ballot data file, and the Web manager wrote the perl programs that ran the voting process. Communication between these groups helped ensure a smooth election process.

While the techniques we used might not be appropriate to institutions with different technological or student environments, they were well-suited for us. The online elections helped showcase our technology while making the election process much easier for students.

Paul Dempsey (dempsey@dickinson.edu) is Web manager at Dickinson College.

Endnotes:

1. Federal Election Commission, "National Voter Turnout in Federal Elections: 1960–1996." [www.fec.gov/pages/htmlto5.htm]
2. "Student Elections at Haverford College Attract a Big Turnout Online," *The Chronicle of Higher Education*, Feb. 18, 2000, p. A60.
3. Butch Oxendine, "Electronic Elections," *Student Leader*, Spring 1999. [www.studentleader.com/sl_10ee.htm]