Joey Chau
Mark Dlugokencky
Lauren Dana Rosenblatt
Kevin Sheu

Security and Privacy in Computing 600.443
E-Voting Project – Part II

Registration

We will work with the assumption that the voter registration system has already been set up. Voters will be given a pin number upon registration to vote. This pin number, along with the last four digits of their social security number and their zip code will be used to authenticate themselves at the poll sites. Upon, authentication, a new number will be given to the voter, thus removing any connection between the voter and his/her vote once it has been cast. The voter will use this number, along with their zip code to place his/her vote.

Ballot Formation

Once the ballot in each county is set, the information will be set to a central location for each state. As soon as this location receives the ballot for every county, it will send the information to a group that will compile the information from all 50 states. A database containing all of this information will be placed on every kiosk computer and its respective county cluster. This will be done to facilitate remote voting during a denial of service attack. We also plan to have a website that will have a sample page for each ballot. A voter may look at the website before voting so there are no surprises at the poll site; the ballot they see there will be exactly the same as the one they use during the actual voting process. They will have the opportunity to view the entire candidate list for their district or any other district from any location.

User Interface

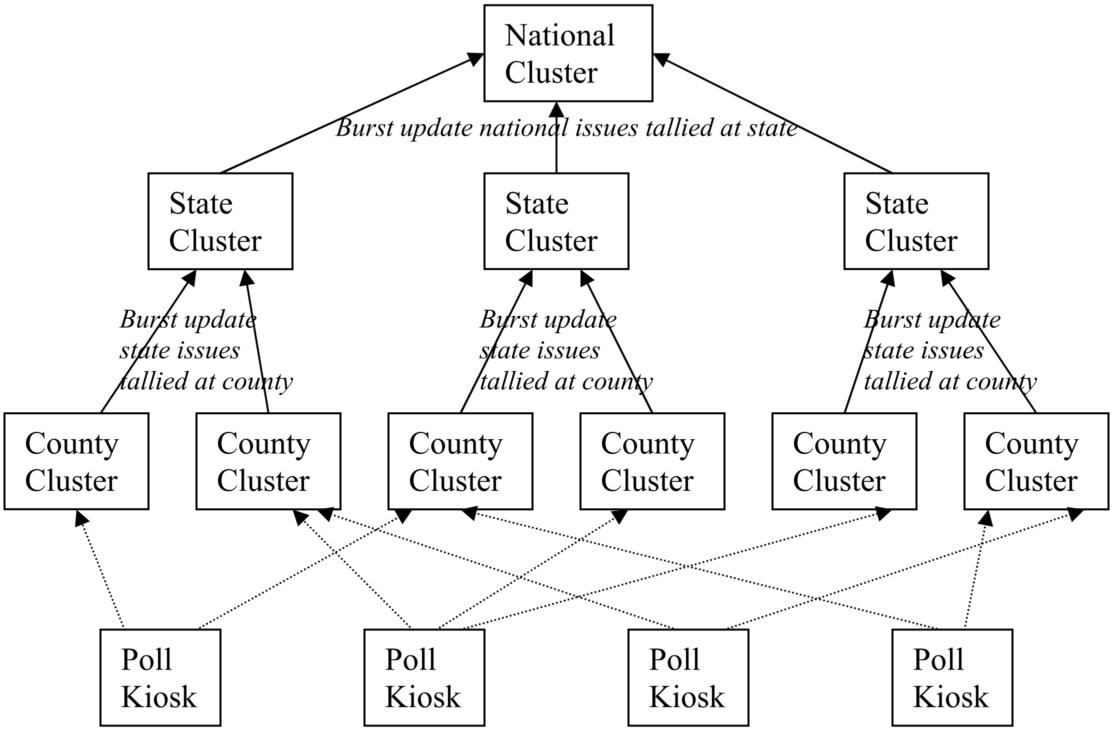
Our system will have a simple, easy-to-use interface. It will contain straightforward pages with instructions telling the user exactly what to do at each stage. There will be textboxes for the voters to put in their information, as well as submit buttons to send the data to the proper location. The voter will vote for each office on a separate page, so that all information can fit on one screen. We do not want the user to not vote for a candidate just because he/she did not see the candidate listed and did not think to scroll down. The pages will have a circle (or some type of object) next to each name. The user will click appropriate one to place his/her vote. There will be no ads or irrelevant text to distract a user while voting. The pages will only have election-related material on them.

Software Detail

The software will be developed in java. The client software will be composed of two components. The first program will be for authentication, and the second will be for the actual voting. Users will input their zip code, pin number (received during registration) and the last four digits of their social security number. They will then be given a random number. They will open up the second program and type in the random number (which proves simply that they are a registered voter) and their zip code (so that they receive the proper ballot). The data is stored in two separate databases: one to hold the voters and one to hold the votes. There will not be a way to connect the voters to the vote they made, thus providing evidence to the voters that their vote is in fact private and cannot be traced back.

<u>Hierarchical Structure</u>

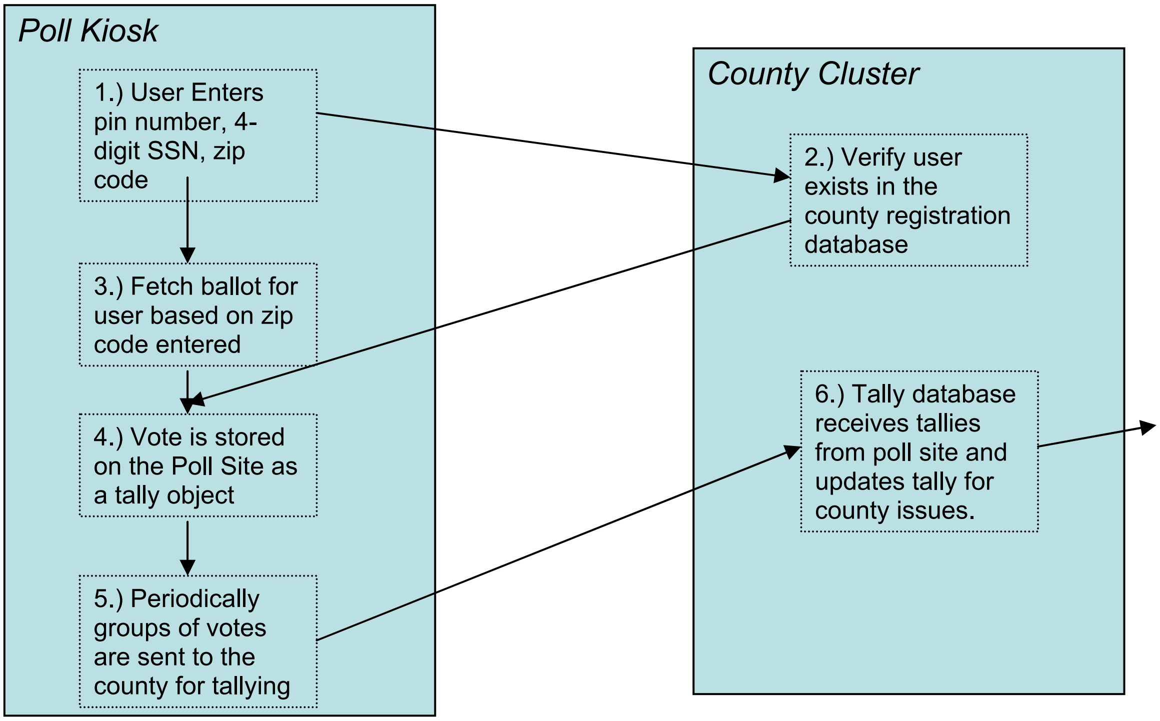
In our design, we plan to implement a hierarchical structure to our voting system. Each layer of the system will have its own cluster of independent distributed server systems. The cluster will be used as a mechanism to both provide backup and to protect against denial of service attacks by providing multiple points of information access. Each cluster will contain between three and five computers/servers. The servers within a cluster will maintain a database for registration. In addition, the servers will maintain a data file of all the vote tallies that have occurred. This will be detailed in a later section. Each cluster will only tally the information pertaining to issues at that level. It will then only pass on vote counts and not individual votes. For example, the county clusters will tally all issues that pertain to city council elections. Votes for state representative will be stored on the county clusters and then sent up to the state cluster where the votes will be tallied. Similarly, votes for president are passed up from county level to state level and then eventually passed up to the national cluster. Below is a diagram of the hierarchical structure.



Kiosk level

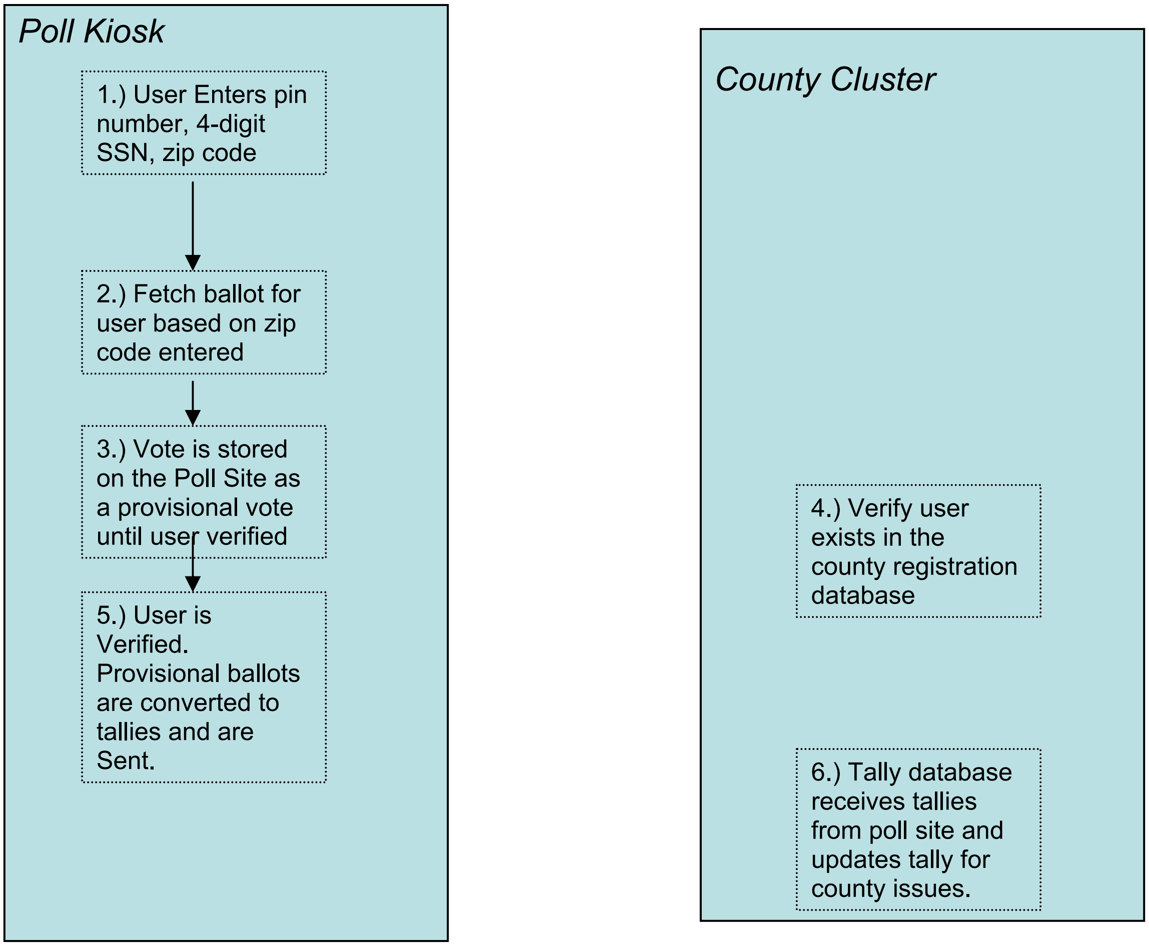
Kiosks are light weight voting entities that contain a voting program, a ballot database, and a storage database for tallies. When a user logs into a kiosk, they provide the necessary information – pin numer, 4-digit SSN, and zip code. The zip code is used to find the correct county cluster to ask for verification. Furthermore, the appropriate ballot for the designated zip code will also be queried from the poll kiosk and presented to the user. The user votes and this information is stored with their identification as a provisional ballot. If the county verifies the user, then the user information is stripped from the tally and the tally is saved and waits to be transmitted to the county level. In the case of a denial of service attack where the county database cannot be reached, then all user login information will be stored at the local kiosk machine, and the voter will be given a provisional ballot. Once the kiosk machine is reconnected to the County Cluster, the registration information will first be verified, and once this has been completed, the Provisional Ballot will be converted to a tally and sent to the cluster. In the case of a loss of connection with the County Cluster, a time period, T, will be defined for how long a time period to wait before attempting another reconnection.

Kiosk Communication with County Cluster During Normal Operation:



Poll Kiosk
1.) User Enters pin number, 4-digit SSN, zip code
3.) Fetch ballot for user based on zip code entered
4.) Vote is stored on the Poll Site as a tally object
5.) Periodically groups of votes are sent to the county for tallying
County Cluster
2.) Verify user exists in the county registration database
6.) Tally database receives tallies from poll site and updates tally for county issues.

Kiosk Operation with County Cluster when Communication is Blocked:


Poll Kiosk
1.) User Enters pin number, 4-digit SSN, zip code
2.) Fetch ballot for user based on zip code entered
3.) Vote is stored on the Poll Site as a provisional vote until user verified
5.) User is Verified. Provisional ballots are converted to tallies and are Sent.
County Cluster
4.) Verify user exists in the county registration database
6.) Tally database receives tallies from poll site and updates tally for county issues.

<u>County, State and National Levels</u>

Periodically, votes will be tailed at the county level and then forwarded to the state clusters to be added to those counts. We chose to do this rather than just send the final counts upon the close of elections. If we waited till the end a denial of service attack could make it so no votes make it to the state level. This way, a large percentage of the votes will already be there. In the same manner, vote counts will periodically be forwarded to the national level. This will be very beneficial to news stations that like to keep up-to-the-minute accounts of how the election is going.

<u>Remote Voting Detail</u>

There must be a way for each person to be remotely authenticated. Remote voters will use the exact same interface as local users, but their zip code will be identified by the system as being out-of-county. It will then be transmitted to the proper county for verification. Upon verification at the correct poll site, the voter's zip code will be used to identify the county in which the voter is registered to vote. The voter will then make his/her vote, and it will consequently be transmitted to the proper county's server cluster. In the case of a denial of service attack, the votes will be stored and sent out once the denial of service attack is resolved. Backups of all remote votes will be stored at the polling site at which they were made.

Server Cluster Architecture

Server Assumptions

- As mentioned earlier, we are assuming that the registration process has already been completed. Our design focuses on the server architecture during the actual vote casting and counting process.
- We assume that there are no network partitions.

Overall Design

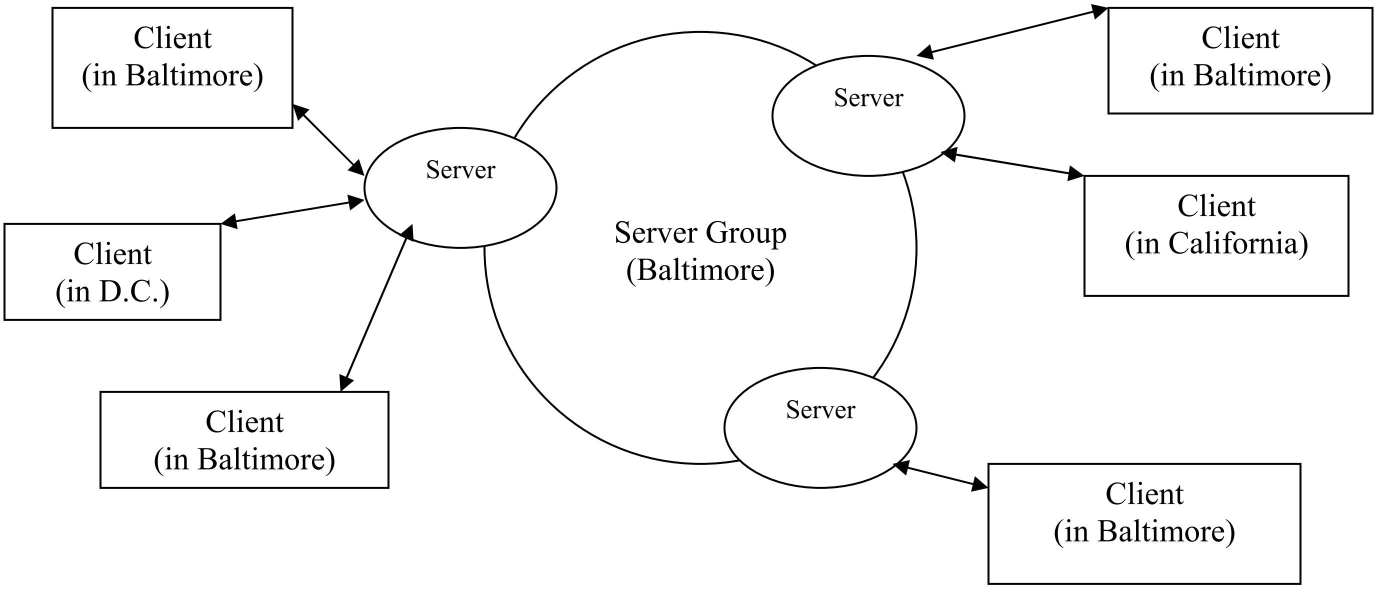
We have designed our system so that every district/poll site has a cluster of three servers. A voter can connect to any machine within any cluster, so that people can vote from any poll site in the country, and do not have to be physically present in their home district. The connection process will be detailed later. The additional servers provide reliable service and increased availability should any server fail. In this section, we introduce a unique protocol so that the three servers provide:

1. Consistency between data on the servers
2. Redundant and Reliable vote and registration storage
3. Availability and service remains unchanged as long as at least one server is available. It is persistent in light of server failures.
4. Load balancing between the servers, so that users will not have a long delay when voting.

Although it was not the original intention, the protocol provides for some protection against Denial of Service Attacks. Should a server come under a denial of service attack, the client machine can switch to another machine. However, in light of the limited number of machines, this policy becomes exhausted once all three servers have failed. However, our system has accounted for denial of service attacks at the client level as well by only sending out votes when the servers are available.

Each server group will communicate between servers through a multicast protocol. In the prototype, we have chosen the Spread Toolkit (introduced in Distributed Systems, 600.437) to provide the reliable multicast service.

The system architecture is designed so that all servers in a cluster are members of one multicast group. The communication between servers is completed through the server group; the server communication protocol will be detailed later in the document. Additionally, each server is a member of a personal group. The personal group only has a single member, and nobody else can ever join that group. The client communicates with a server by directly sending messages to the personal group of the server.



Server Design

The basic concept of our servers involves a responding system which maintains vote information. In general the server never initiates any type of information transfer except for initial start up message. The core of the server is a system of maintaining two crucial data structures: the vote list and the action queue. Together, the two data structures work to preserve voting information and ensure reliable consistent communication of voting action. Secondly, servers maintain knowledge of which other servers in currently in its group.

Data Structures

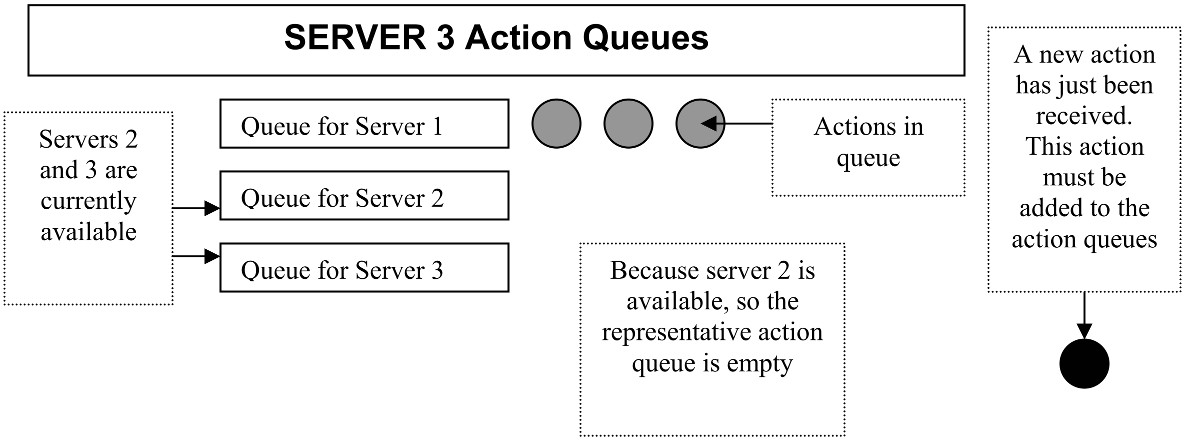
The server maintains two sets of data structures to provide reliable and consistent communication. The Vote List, contains a list of Vote Issue Objects, which each contain an issue that is being voted upon, and maintains a tally for each issue. As an example, in the 2000 presidential election, a Vote Issue Object would contain:

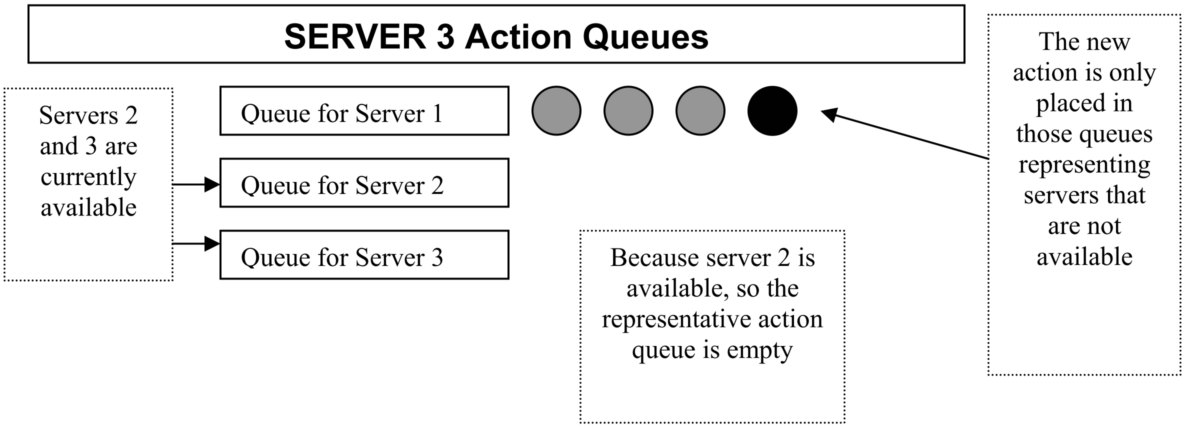
President of the United States	
George W. Bush	321
Al Gore	325
Pat Buchanan	11

Each time a Vote Issue Object has been modified, which happens during a new vote, the queue is written out to disk. Thus, in the event of a server crash, or Denial of Service attack, the information can immediately be recovered upon server revival. In addition, if the server cannot be brought back up, the file is portable, and can be moved to a separate machine.

As a note, when a voter votes, his or her registration information is also updated. It will indicate in the database that the voter has already voted. Therefore, if the voter tries to log into the system again, the database will recognize that the voter has already voted, and prevent him or her from voting a second time.

The second data structure that a server maintains is a set of action queues. On each server, there is an action queue representing every server – because we only have three servers, every server has three action queues. The concept of an action is any transmission on the server communication group that affects the vote tally. Anytime a transmission which alters a vote is received, the server will transform the transmission into an action – an action is a representative container of the transmission and the transmission type – and place the action into the action queues. Not all action queues add the latest action. Depending on which servers are available – as described in the membership maintenance section – only those queues representing servers that are not available will add the current action. The result is that on every server, there is a queue of actions for every server. For a given server A, the action queues representing servers that are available will be empty. For the same server A, the action queues representing servers not available will hold all the actions that those partitioned servers have missed. When a server determines that a server has become available again, it will flush its queue for that newly joined server. Flushing the queue will bring the new server up to date. It is important to note that if there are multiple servers in a group when a new server joins, only one of the grouped servers will actually transmit the flushed actions.


SERVER 3 Action Queues
Servers 2 and 3 are currently available
Queue for Server 1
Queue for Server 2
Queue for Server 3
Actions in queue
A new action has just been received. This action must be added to the action queues
Because server 2 is available, so the representative action queue is empty


SERVER 3 Action Queues
Servers 2 and 3 are currently available
Queue for Server 1
Queue for Server 2
Queue for Server 3
The new action is only placed in those queues representing servers that are not available
Because server 2 is available, so the representative action queue is empty

Server Availability Maintenance

Each server maintains availability information about the servers in its cluster. This is represented as an integer array of size three. Each slot represents one of the servers, and a value of zero indicates that the associated server is not in the partition, and a value of one indicates that the associated server is currently in the server. Every time the server receives any type of availability information, this array is updated. At the same time, whenever a server receives availability information, it also determines if there are any new members have joined the group or partition. One machine from each partition or group will then update the new members that have joined this group.

Initialization and Maintenance

Before servers are placed in the state where they can communicate with one another, the server must first initialize its data structures. This mainly involves reading data files currently stored on the machines. Every server has a Vote List data file that stores information about all the information in the Vote List. Also, every machine has three Action Queue data files for each server in the cluster. The Vote List must be loaded as well as the Action Queues because if a server is brought down, other servers will only stored actions occurring after the crash. In order to preserve consistency, servers must be able to return to the state right before the crash. Furthermore, storing and loading action queues is necessary to allow downed servers to actually update new servers if it boots up in a different partition!

Client

Before each communication with the server, a client machine will randomly choose one of the servers within the cluster group for the specified district. A client will initially check to see if the server is running. It is essentially a ping, where the client sends a zero sized packet to the server, and waits for that zero sized packet to be returned. If the server does not return the packet, then it is an indication that the server has failed, and the client is switched to one of the other servers.

Client Action on Server Failure

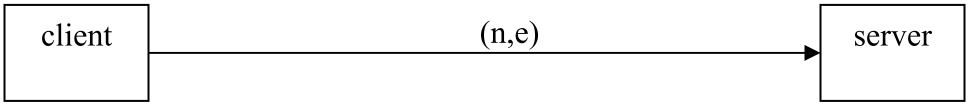
If a server becomes temporarily unavailable, the client will be able to log into a separate server. The communication between all the servers that has been detailed will provide the same service to the client. This provides increased availability in light of server crashes, or limited Denial of Service resiliency.

Encryption

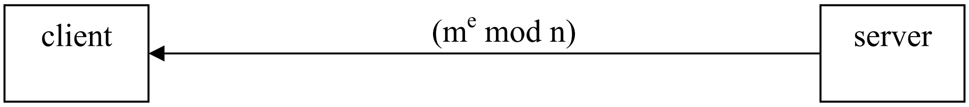
After the user enters in their three unique numbers, they will be encrypted and sent to a server for authentication. The server will then send back an encrypted response and if the person is verified, a random number for use in the second program. In the second program, encryption will be used in the same way. An encrypted copy of the vote and corresponding zip code will also be printed, thus ensuring security of the vote.

First, we need to exchange a public key with the client (Pole Sites) and the server (County). For this we will use 1024-bit RSA encryption.

Client sends n and e to server

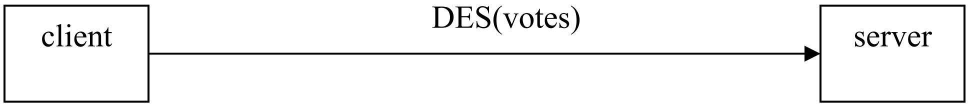


Server chooses random number by a “random” number generator and encrypts random number with e.



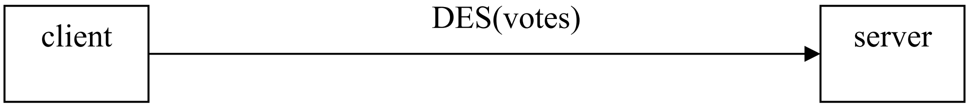
Client decrypts by using a 56-bit DES block cipher. We chose this because it can encrypt fast, and cannot be broken in one day.

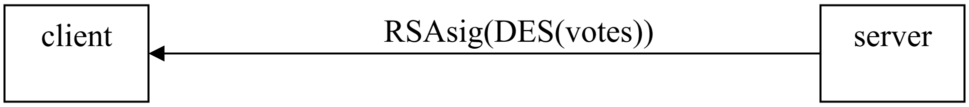
Client encrypts voter results with DES and sends packets to server



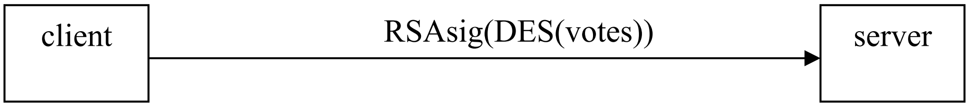
Authentication

The routers will know where the packets came from and thus can reject any packets that have not come from assigned IP’s to client poll sites. The packets need to be signed by the server so that the client knows that the actual parent server received these packets. Therefore, as the server receives each packet it will sign it and return it to the client.





Now, the client can be sure that the proper server is receiving the packets. Next, we must verify to the server that the packets received are from a legal client. Therefore, the client will sign the same message (DES(votes)) as the server. And the server will verify.



This will ensure that each packet that arrives at the server is from a legal client. It will also ensure that a legal server received each packet sent. This is modeled after the three-way handshake that occurs in a TCP connection. Since there will not be a large amount of packets there will not be a significant delay.

Duplicate Votes

In our current voting system, a voter may only vote once; voters go to the poll site, show proof of identity and then place their vote. With an online system, things are not as easy. We would like to ensure that the same is true of our system. Once authenticated, a Boolean variable will be set saying that the user has voted, and thus cannot vote again. In the case of a DoS attack, a user could vote at multiple poll sites in an attempt to cast multiple votes. To prevent this, the first vote will be the only vote counted. Most of the time, the first vote cast will be the first vote to be authenticated, although there may be a few rare times when different servers go back online at different times, thus causing a different vote to reach the correct distraction first. This will also prevent voter coercion as the voter cannot prove who he/she voted for and can only vote once. Thus, a voter cannot keep switching their vote in order to make money from opposing sides.

Paper Audit Trail

An anonymous copy of each vote will be printed in encrypted form and stored in case of an audit. We have decided to design our own prototype for this. It will allow the user to view a paper copy of their vote, but not allow it to take the copy with them so as to avoid coercion, since the voter cannot use it as proof of who they voted for. It would be similar to a cash register receipt. It would print the vote onto the small roll of paper, which would be encased in a box. The box has a glass window that is the exact size of a printed ballot. Upon voting, the voter's ballot is printed out, and he/she may view it (voters who are not computer-literate will feel better about their vote after seeing it on actual paper). The glass will somehow get covered until the next person votes and his/her ballot is printed.

Design Tradeoffs and Benefits

In Part I of the assignment, we stated our main goals to be: accuracy and integrity, eligibility, authentication, uniqueness, provide an audit trail, convenience, simple and unambiguous user interface, cost-effective, inclusion of remote voting, and separation of voter and vote. We feel that our design covers each of these aspects to some extent. A user will be authenticated and all of his/her information will be kept private and separate from their vote. The system will have a clear interface that will be easy for the user to vote with. It will have everyone on one screen so the user will not have to scroll and thus miss important candidates. Our idea for a paper audit trail as well as having clusters of computers instead of single terminals will help insure multiple ways for an audit if necessary. We hope that our system is cost-effective. We do not anticipate the cost of servers/computers to be too much and the printer device should be pretty cost-effective as well. The majority of the software and databases will be the same for every computer, thus the development costs will be one-time. We have not yet decided on a solution for people with disabilities.

Our security requirements included: user authentication, client server communication, encryption, and prevention of malicious payload, denial of service, fraud and coercion. We focused the most on providing user authentication, encryption, and preventing denial of service attacks, fraud and coercion. Our interface authenticates a user without having the user give out important personal information. The encryption schemes we have chosen will provide adequate protection of our data throughout all communications. Since we cannot protect against a denial of service attack, we found ways to have the system work even when it cannot connect to other servers. As long as they all get online eventually, there will be no data loss and all votes will be counted. Our duplicate vote and paper trail features helped ensure faith in computer illiterate voters as well as protect against coercion.