People Power
FREE: Free Referenda & Elections Electronically
- Technical Documents -

by
Jason Kitcat
1999 − 2000

Supervised by
Dr Iain Munroe & Dr Jane Sinclair
University of Warwick

Keywords

democracy, electronic voting, Internet, Java, networks, privacy, security

Abstract

After providing a broad analysis of the subject area and motivations for the work, the current state of theoretical work on electronic voting is reviewed. Working from these concepts the author describes and explains a novel design for an Internet-based electronic voting system using Java. The implementation of this system is then explored and any problematic issues arising are swiftly dealt with. The document ends with a brief summary and a review of the new avenues of exploration that this research has exposed.

Contents

1. Introduction

During previous research on the changing nature of conflict due to the Information Revolution (InfoRev), it became apparent that an interesting area of study was the increasingly influential part civilians were playing in global issues. For example their previously unthinkable ability as part of Non-Governmental Organisations (NGOs) to not only instigate the Rio de Janeiro EarthSummit but to also use the event to *negotiate* with national governments is a remarkable indicator of their increasing power.

Having started some preliminary research it was clear that the role the InfoRev played in this empowerment was undervalued. This is a subject area which is extremely relevant to the everyday life of all people — how we live today can be directly related to the actions of early instigators of people power such as Mohandas Gandhi and Rev Martin Luther King. Our lifestyles are constantly being shaped by the interplay between interest groups in government, the influence of the corporate world and the cries of NGOs. The process is continual as today what we eat and do is being affected by groups such as Greenpeace acting against genetically modified organisms, Amnesty campaigning for freedom of information and other NGOs' campaigns.

We are also seeing an increasing number of ad hoc coalitions forming around single issues representing a wide variety interest groups. This can result in extremely powerful and explosive forces as demonstrated by the Seattle WTO protests. My research and resultant thoughts on this topic area can be found in the accompanying work, *People Power: The Revolution in Civilian Affairs within the Context of the Information Revolution and its impact on the Political Process* (2000). I will avoid repeating what is said in the accompanying document which analyses all the issues in great detail and thus I will focus solely on technical matters.

In studying this subject area I became painfully aware of the repeated declarations that InfoRev technology would create vastly strengthened democracy by allowing the electorate to vote with greater ease and more often. However electronic voting has not arrived in any great force nor has any other part of the democratic process (such as public discourse) been greatly facilitated by technology. There have been tentative first steps for fifteen years, but none have proceeded beyond an experimental stage.

Currently there are several systems being developed to enable electronic voting via the Internet. The emergence of the Internet as a globally available communications infrastructure has radically eased connectivity and access issues that dogged previous systems, such as Qube, which were reliant on cable TV thereby providing a much needed boost to this subject.

The UK company Entranet has presented a system for use by a small number of UK local authorities authorised to pilot electronic voting. According to reports on the BBC's Tomorrow's World and *Computing* the system's key innovation is that It can be used on a wide variety of Internet-enabled devices ranging from Palm computer to WAP mobile phone to WebTV. However this, and all other Internet based-systems, have some key failings which I will address in the Design & Analysis section.

Thus I had identified the unfulfilled promise of the InfoRev to improve democracy, which even the latest developments did not seem to fully address. The opportunity for the populace to be more involved in the

running of their country and thus the possibility of a more responsive government presents one with an extremely desirable goal. Therefore I decided that an electronic voting system for use on the Internet would be created, hoping that it could help jumpstart e-democracy while addressing the failings of contemporary systems.

2. Design & Analysis

Failings of Current Systems

Probably the earliest form of electronic voting is via the phone. It has been used in pilot projects in some local elections and notably by the Labour party's National Executive Committee. The system works by each user being issued a PIN number and ID number which is then used to access a phone menu system (as used in many business applications) to allow the user to select their party of choice. Phone systems are highly usable and with an installed infrastructure that is available to most, if not all, voters in the Western world.

Additionally, once the systems are in place and the investment has been made to provide users with their login details, such systems can be expanded to provide other services such as driving test reservations and tax payment. Unfortunately the biggest problem with any phone-based system is also voter authentication. PIN numbers used for login tend to be 4 digits long and any hacker of medium competence could create a system to churn through all possible combinations to gain access. Additionally there is a burden in distributing these PINs and ID numbers to every voter, and in the process this exposes them to potential malicious access while in transit.

Some progress on phone systems was made with the very early interactive TV pilot projects such as Qube in the USA which used the increased bandwidth available in cable TV to improve the usability and attractiveness of the systems, but the basic technology still relied on phone-style number pushing and thus the same security issues weren't addressed satisfactorily. One positive note is that cable TV networks tend to be more physically secure than phone network cabling due to the fact that TV companies buried their cable. However cable installation is an extremely slow process which has made availability an issue. Additionally every cable provider uses a certain amount of proprietary systems making scalability and national coverage extremely difficult.

Optical vote reading techniques developed at approximately the same time as cable unfortunately don't offer any real changes, just more accurate and speedy vote counting compared to paper-based predecessors. However in an article[i] comparing the implementation of a hybrid optical/electronic system in Norway and the use of fully electronic, but specially designed and thus expensive, voting booths in the US both systems failed to perform satisfactorily. Major issues were a lack of sufficient testing and the reliance on proprietary technology for the systems, leaving authorities locked into expensive maintenance and upgrade cycles.

The electronic system in the US was based on booths. Several companies offer these booths, but none are interchangeable and thus a customer must commit to buying all their systems from one supplier. These booths suffered from many complaints including a lack of privacy (resulting in some voters bringing

umbrellas to hide their choices!), difficulties in configuration and problematic maintenance and improvement due to their proprietary nature.

The article [Lar99] concluded by laying out a set of principles for all voting systems which match those which I identify for FREE later in this section.

More recently with the advent of the Internet, e-mail has been used as a voting tool. A large number of free programs are available to automate this process and count the results after e-mails have arrived. The major benefits of such a system are its low cost, simplicity and thus ease with which it can be implemented. However it is extremely insecure, there is no way to prevent votes from being tampered with, and few systems can successfully prevent 'vote spamming' where one user may send multiple votes in to try and swing the result. Similar comments apply to web-based straw polls as seen on many news web sites, the only added benefit they offer being enhanced usability over the text-based e-mail systems. However both web and e-mail based voting systems are available at no cost thereby enabling any organisation with Internet access to perform votes in a quick and easy manner. But their dire lack of security makes such systems totally useless for votes of any importance.

The newly launched companies hoping to cash in on the massive economic growth seen in the Internet market, as well as the ever growing amount of money being associated with politics, unfortunately don't seem to offer any particularly novel ways of addressing security issues. Most rely on the world wide web, which admittedly has become a model of ease-of-use and something most voters will be familiar with, putting them at ease with an otherwise totally novel experience. Entranet, Votation.com, VoteHere.net and eBallot.net all appear to rely on the Secure Sockets Layer (SSL) encryption system[ii] to secure all communications between client and server.

This standard is undoubtedly secure, it is used by all major e-commerce sites to protect credit card transactions, and is supported by banks who are notable for their detailed attention to security. Of course it will need to evolve as computers do to keep the encryption hard enough to crack to not be worth it. Also SSL usually requires the purchase of certificates which currently cost over $150 each, unacceptably raising the barriers of entry. Most importantly SSL does not secure either the client or the server, only their communication channel. There have been several server-side vulnerabilities recently exposed that enable a hacker to access data once it has been decrypted from the SSL stream. Additionally web servers are a centralised system which make them a glaring single point of failure therefore rendering whichever server hosts an election a prime target for terrorist attack. Such an attack could wipe out all vote data and prevent the election from functioning at all.

Perhaps the most salient point with these corporate systems is that none of them have addressed the voter authentication issues adequately. Most are reticent on what they precisely do, but it appears they rely on giving each voter a PIN number via the web which can only be used once. While this does compartmentalise the amount of damage that a hack can do, it is not a satisfactory solution to a serious issue. In fact the system proposed for use in Arizona later this year requires users to register at a web site which then dynamically delivers them login in details for voting. Such automated and real-time systems do present the possibility of some extremely effective hacks whereby people with malicious intent could attack the system to spew out vast numbers of valid voter logins. Thus vote security itself is not attacked but the results, while seeming valid, are thrown.

It would seem a logical way to approach attacking such systems — a lot of time and publicity is spent on reassuring users of how secure making a vote is — while little time is spent closing easier loopholes. I propose to address all these areas of concern with regards to security. As part of the development of FREE a review of the current research on the subject area was conducted the salient points of which I shall highlight.

Theoretical   Techniques

A large portion of the research in this area is actually driven by the desire to make <u>the</u> secure and reliable electronic payment system of the future. However techniques relevant to electronic payment generally apply to voting as well. Sending data that represents money in a secure manner is indistinguishable from sending data that represents a vote.

As is to be expected most of the research involves incremental improvements on existing well known security schemes. The two more novel concepts proposed recently are by J.C.Benaloh and D.Chaum.

J.Benaloh proposes [Ben96], through the use of some complex math, a highly distributed system (and thus robust) for private and verifiable electronic voting[iii]. His concepts rely on what he calls 'tellers' to count votes in such a way that the results can be verified without anyone having to know who voted what. By using a large number of these 'tellers' he argues that any such system would be extremely resistant to outages and attacks. Other areas of Computer Science support this view, take for example the development of web serving techniques which are becoming reliant on parallel processing (local area distribution) and load balancing systems (regional area distribution) which direct data from servers chosen with algorithms which select servers according to the criteria of quantity of free resources and physical proximity to the client.[iv] Thus I have felt it prudent to use such ideas in the design of FREE to increase it's robustness.

Mathematically proven as verifiable and private, Benaloh's system provides a viable alternative to the paper ballot. However in his thesis *Verifiable Secret-Ballot Elections* he himself concludes:

> 'It does not seem likely that the methods for holding a verifiable secret-ballot election presented in this thesis will be used in actual elections in the near future … public acceptance and the wide-scale availability of the required technology seem to be some distance away.'[v]

David Chaum is more optimistic, and has been working hard at DigiCash and the Amsterdam Centre for Mathematics and Computer Science to bring his ideas for a private system encompassing money and voting, as expounded in the seminal *Security without Identification: Transaction Systems to make Big Brother obsolete* and other associated papers.[vi]

Essentially this system [Chaum85] allows the use of separate pseudonyms for each organisation a person interacts with. These pseudonyms cannot be reconciled to create a vast image of the person's activities by collating data from separate organisations. This is because each organisation only can access data related to their particular pseudonym for that person — the user controls all these pseudonyms from a smart card or handheld computer, this being the only point where all the pseudonyms come together. Such a system allows large-scale automated transaction systems to protect privacy and maintain security for individuals and organisations. So, while an organisation can build up a picture of your pseudonym's habits in its

interactions with that particular organisation, it has no way of resolving this with your dealings with any other organisations. It is conceivable that in Chaum's world marketing organisations might offer to pay you for access to some of your pseudonym's so that they could resolve your actions into a single profile – but this will be an opt-in alternative, not the opt-out world we are currently in.

Chaum's system is technically simple and thus entirely feasible which makes it an extremely exciting possibility. Chaum is now totally focussed on electronic payment systems and no longer spends time developing his concepts for use with electronic democracy but this shouldn't discourage its implementation. Perhaps its use is discouraged by the need for all organisations we interact with to implement this system before it will truly be worthwhile, such explosive adoption is extremely hard to achieve.

But the key problem with this concept is certainly its dependence on a handheld computer or smart card being available to every user as a way to manage these numerous pseudonyms. Not only is losing such a central item problematic – how do you get all the information back if it's so highly compartmentalised that nobody knows who you are? But also, presently neither smart cards or handheld computers are likely to be anywhere universal in the near future, though this is a distinct possibility in the long-term future. Thus, between difficulties in achieving sufficient adoption between users and organisations, for the moment Chaum's system remains theoretical.

In March 1999 the Secretary of State of California created the 'Internet Voting Task Force' which has developed some interesting conclusions [And00]. The leader of the body, Alfie Charles, states that the body has uncovered some major impediments to voting online that will not allow it to completely replace traditional voting. The task force has serious doubts over protecting individual voter privacy, security as well as the integrity of a ballot Other primary concerns are:

- An inability to identify online voters with certainty.
- No guarantee of the security of remote computers, especially from 'Trojan Horse' attacks which might allow other users to gain control of voters' computers (cf. "Back Orifice")
- The inability to guarantee privacy on office computers where network administrators have high levels of access to users' computers and might be able to examine unencrypted voting information on users' machines.

But despite all these reservations, they have described an extremely sensible four-part transition to allow limited voting online. The first step will be the introduction of computer voting in locations controlled by county voting officials with the eventual goal of enabling voters to cast votes from any remote location.[vii]

So, having digested the more theoretical ideas and also accepting the conclusions of the Internet Voting Task Force I have proceeded to lay out some key principles for an electronic voting system which I used to guide the FREE development process.

What is needed

Taking heed of such privacy and security issues along with the lack of widespread enabling technologies such as smart cards, I created a set of directives through which all decisions regarding an electronic voting system would have to be filtered:

1.	Security is essential.

An obvious but necessarily important point. Authorities go to great length to provide effective security with current paper ballots and this must not only be matched but improved upon. This is a particularly tall order considering the existing poor record computers hold with regards to security. Public confidence in the validity and security of a ballot are essential to attaining sufficient turnout as well as achieving significant confidence in the results of any vote.

2.	Voter rights, especially privacy, must be maintained.

Computers have gained an equally poor record with regards to security in recent years with the discovery that many companies use their software to track their customers' behaviour – e.g. RealPlayer, CometCursor and DoubleClick have all been discovered surreptitiously following Internet users' every move. This image must be countered in a convincing manner, otherwise it is understandably unlikely that people will vote with a system where they feel unsure as to who will know what they voted. Any system which provides a chink in the armour of privacy is open to abuses, such as coercion into voting for a particular party.

3.	Openness and trust are essential to any system.

This is applicable to any system, not just election systems. An excellent example is the Linux operating system where all the source code is freely available and modifiable. Thus any user, should they want to, can understand what Linux does to their computer, how this is done and can thus alter this to better suit their requirements. Openness instinctively breeds trust by indicating that there is nothing to hide. Additionally open source software (such as Linux, Apache or Mozilla) tends to be more robust and stable as anyone in the world programming community can contribute their ideas and talents to the code base.

4.	The system must be fast and survivable.

Open software breeds these characteristics which are essential if an electronic system is to ever replace paper-based systems. The key benefits of electronic voting are count speed, accuracy and low cost. But for the average user ticking a box on a paper slip is fast and intuitive. Additionally, due to the regional nature of ballot counts, the current system is robust in the face of attack or malfunction. Any computer-based system must rival this and must not provide opportunities for criminals, terrorists or any other group to undermine a vote. Speed also relates to usability – a more responsive system is more useable.

The FREE design

So how was this implemented in the FREE system? Here are the attributes I stated above to be key to a successful election system, I will now explore how I addressed these issues.

1.	Security is essential.
2.	Voter rights, especially privacy, must be maintained.
3.	Openness and trust are essential to any system.
4.	The system must be fast and survivable.

In response to rules 1 and 2 a credential system based on Chaum's work [Chaum85]is being used to ensure voter privacy and untraceability while also helping to prevent fraud and vote rigging.

The user identifies themselves to the voting system with a smart card, retina scan, finger print scan or some other very secure system – this is used as the login ID for the system. Before access is granted the ID is verified with an independent organisation I will call Z which is running the ERServer (FREE's Electoral Roll Server) software. All Z has stored in its computers is the identifying feature (fingerprint, smart card key, retina shape), its associated numeric pseudonym and a credential to show whether this pseudonym is a valid voter. If so access is granted to the user.

The user then votes using another pseudonym, called B, which is only linked to pseudonym A (which was used to contact ERServer) by the user. This is most easily implemented by using smart cards, but other ways using third parties with strictly implemented blind databases are needed until the technical infrastructure for smart cards is widely available. With B the vote is registered with the regional server software RTServer which knows the vote is valid because access would not have been granted without a valid credential from Z.

The concept is well summarised by a Venn diagram:
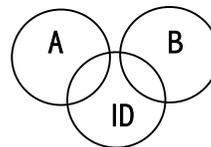
So pseudonym A and B both relate to ID, but cannot be linked between each other. Thus the system records that that a user has voted, but nowhere in the system is a link made between who and what is voted. I have deliberately designed how this is implemented to be extremely open so that if technologies such as smart cards do become more wide-spread they can be used with the minimum of extra work – thereby allowing the system to grow as technology does.

It is worth noting that due to the huge variety of policies on encryption between countries, it is impossible to implement strong encryption without either making the software liable to exorbitant licensing fees and/or making it illegal in several countries. Also worth noting is that encryption systems tend to have a dramatic negative impact on the performance of communications. Techniques for avoiding this are only now starting to emerge from companies such as Intel who are providing expensive dedicated hardware for encrypting/decrypting SSL streams before passing them on to web servers, thereby lightening the load on their processors.

Therefore it is preferable to use pseudonymous systems such as that detailed above and combine this with message authentication codes (MACs) to ensure that vote data hasn't been tampered with. Because a variation of Chaum's system has been used it doesn't matter if a vote can be read as there is no way of tracing who voted, but nevertheless it is important to prevent a hacker changing what a valid user voted.

FREE's MACs are based on message digest[viii] techniques implemented within Java's security framework. In particular the MD5 algorithm (which was created by RSA Data Security) was chosen. Like MD4, the MD5 algorithm is being placed in the public domain for free general use, the MD5 algorithm being a strengthened version of MD4. It has four rounds instead of three, and incorporates other revisions based

on a year's worth of collected comments on the MD4 algorithm. For example, the input access patterns in rounds two and three have been improved, and the rotation amounts have been optimised for maximum "avalanche effect." The additive constants have been made unique in each step, and an additional dependence of each step on the previous one has been added.

To create a FREE MAC the data portion of a FREE packet has a message digest created with the MD5 algorithm. A new message digest is then created of the first digest concatenated with a secret passphrase. This is then included with the packet. On reception of this packet a FREE program (which also has knowledge of the same passphrase) performs the same operation on the data it received and checks to see if the MAC it made matches the one it received. If it doesn't then the packet has been tampered or corrupted while in transit and is ignored.

Additionally all passphrases, server addresses and other initialisation variables are also hard coded into FREE programmes to prevent tampering. Any variables stored in files would be much harder to secure and keep track on. While this approach is not preferred programming practice it does also enable FREE to be 100% pure Java compliant as not all Java clients may have local file systems for storing such files.

The reader may be wondering why more established forms of secure communications haven't been selected as part of the FREE design. There are several good reasons, some of which I have already mentioned. SSL communications are not all that secure and are also costly. SSL, as do most other public key systems, require extensive use of certificates which are costly (and have seen little, if any, use among end users). Encryption systems, public key or otherwise, are also subject to a complex web of regulation as well as expensive and often restrictive licensing terms. Thus, with a commitment to lowering the barriers of entry and opening electronic democracy to as wide an audience as possible, it was felt that choosing such technologies would be counter-productive.
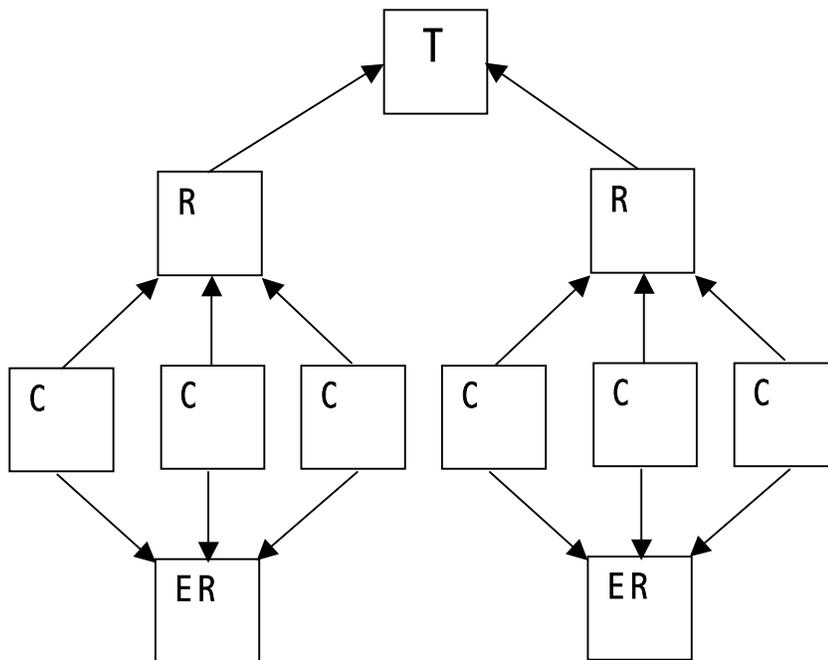
I have addressed rules 3 and 4 by deciding to make the whole system open source software under the General Public License. Thus anyone can use and alter the source code and even sell it, but if sold, the source code must also be included and can still be altered and resold. This means that anyone can look at how the program works, improve it and alter it. I believe this is the only way to create trust in such a system, I am deeply sceptical that voters will trust companies running elections for profit and with systems that are kept secret. This philosophy also aids survivability by getting more people than any company could employ to co-operate via the Internet and continuously improve the program.

There is an issue with this approach: A malicious programmer could alter the code to provide security loop-holes for them to exploit during elections. This is a risk and there are two responses to this. Firstly open source software development uses the 'patch pumpkin' concept, whereby only the one person (in this case the author) who holds the 'pumpkin' can actually make the final changes (or 'patches') to the official version of the software. This acts as a filter to unwanted, malicious or bad program changes. Secondly a guide on how to run an election with FREE will be part of the software, one of its key recommendations will be to get the software audited by an independent technical authority before its deployment for an election to prevent the insertion of any malicious code. While these are not watertight procedures, I would argue that being able to see what the software is doing is preferable to guessing.

In response to survivability and speed the system design is decentralised, thus votes are collected at numerous independent regional servers before being totalled by one central server. Even though this

central server might fail the vote data will be still be kept on all the regional servers — making the overall system significantly more robust than commercial operations relying on single web servers. As an indication of the low level of reliability one might expect from such enterprises, the undisputed e-commerce leader Amazon.com recently admitted in an SEC filing that it has all its systems kept in one leased building with no proper disaster strategy. Reports from many other Internet leaders echo this dire situation which is not only madness for a business totally dependent on technology — but such lack of planning would be an enormous risk for votes of any importance.

Thus, in response to the points above, the overall layout of a FREE system can be drawn as:

```
                           ┌─────┐
                           │  T  │
                           └─────┘
                          ↗       ↖
               ┌─────┐                   ┌─────┐
               │  R  │                   │  R  │
               └─────┘                   └─────┘
              ↗   ↑   ↖                 ↗   ↑   ↖
     ┌─────┐ ┌─────┐ ┌─────┐   ┌─────┐ ┌─────┐ ┌─────┐
     │  C  │ │  C  │ │  C  │   │  C  │ │  C  │ │  C  │
     └─────┘ └─────┘ └─────┘   └─────┘ └─────┘ └─────┘
           ↘   ↓   ↙                 ↘   ↓   ↙
               ┌─────┐                   ┌─────┐
               │ E R │                   │ E R │
               └─────┘                   └─────┘
```

C = Client Machine (in Polling Station)
ER = Electoral Roll server (independent organisation)
R = Regional server (program delivery + vote collect)
T = Totaller server (vote tally collection)

Note that there is no logical limit to how many clients or regional servers are in the system — only computer and network performance limit this variable thus, in theory, a FREE system could scale up to national or global scales.

The choice of Java as programming language was a natural one: I was already well versed in the language, it offers a wealth of networking and security components and its cross-platform (processor-agnostic) capabilities make it ideal for making electronic voting as widely available as possible.

Detailed   Design

The large overall design decisions were then filtered down into more detailed thinking for each particular piece of software.
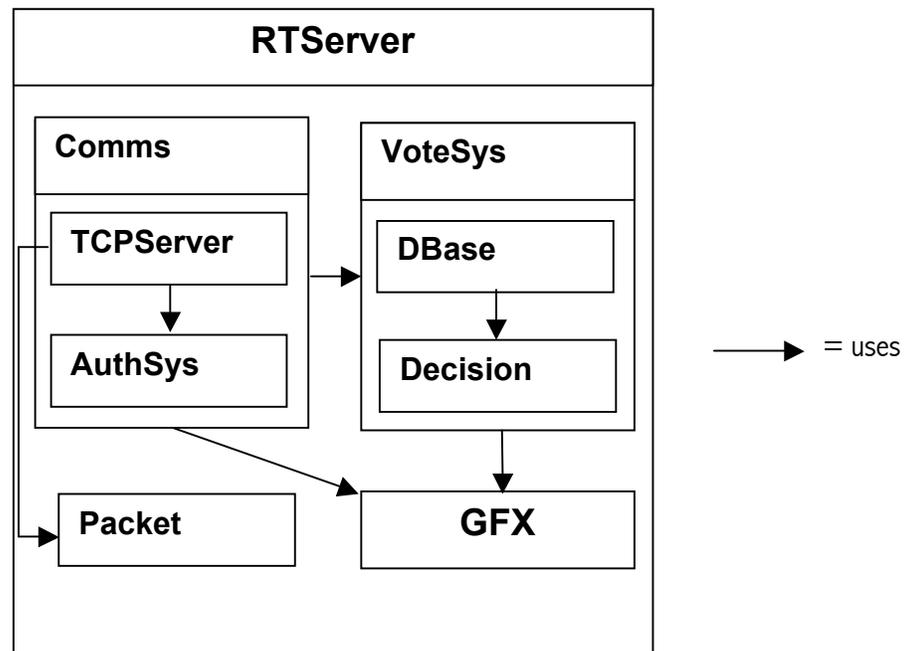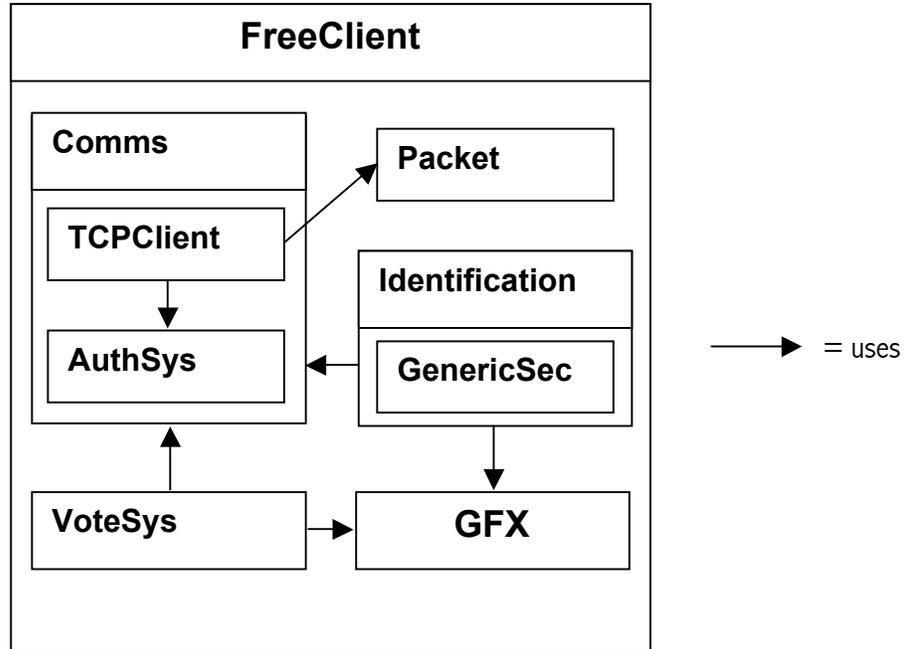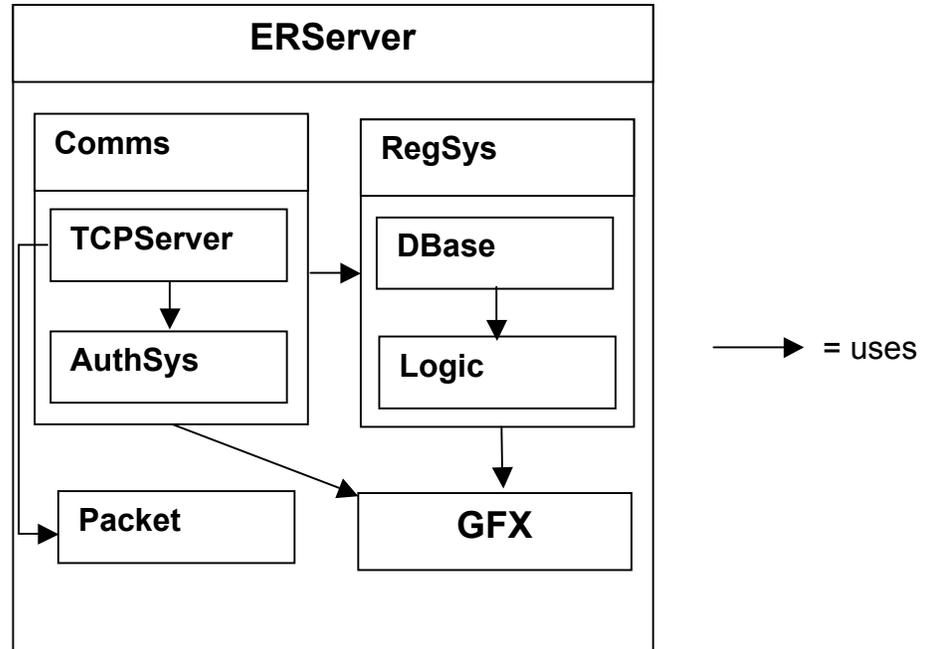
It is apparent that there is considerable similarity between the three programmes and thus shared functionality would greatly ease the programming burden. However while the code would be the same, it would not be shared from the same file – this is a security precaution so that each program will be completely self-contained and as such all security levels can be set to be only within the packages of each program thereby helping to prevent malicious code from accessing FREE's methods.

Note that the **GFX** classes are implemented by Java's Swing packages as **javax.swing** which requires a separate class for every window. Despite this exception, the structure of FREE did essentially follow these designs. Of particular interest is the **Packet** class which implements the data structure for the packets used in FREE communications. FREE packets are defined and formatted as follows:

1|22222-2222222-222222|33333333333333

Where 1 is a single character setting the type of the packet; 2 is the data region of unlimited length with fields separated by a hyphen and 3 is the MAC used to prevent message tampering.

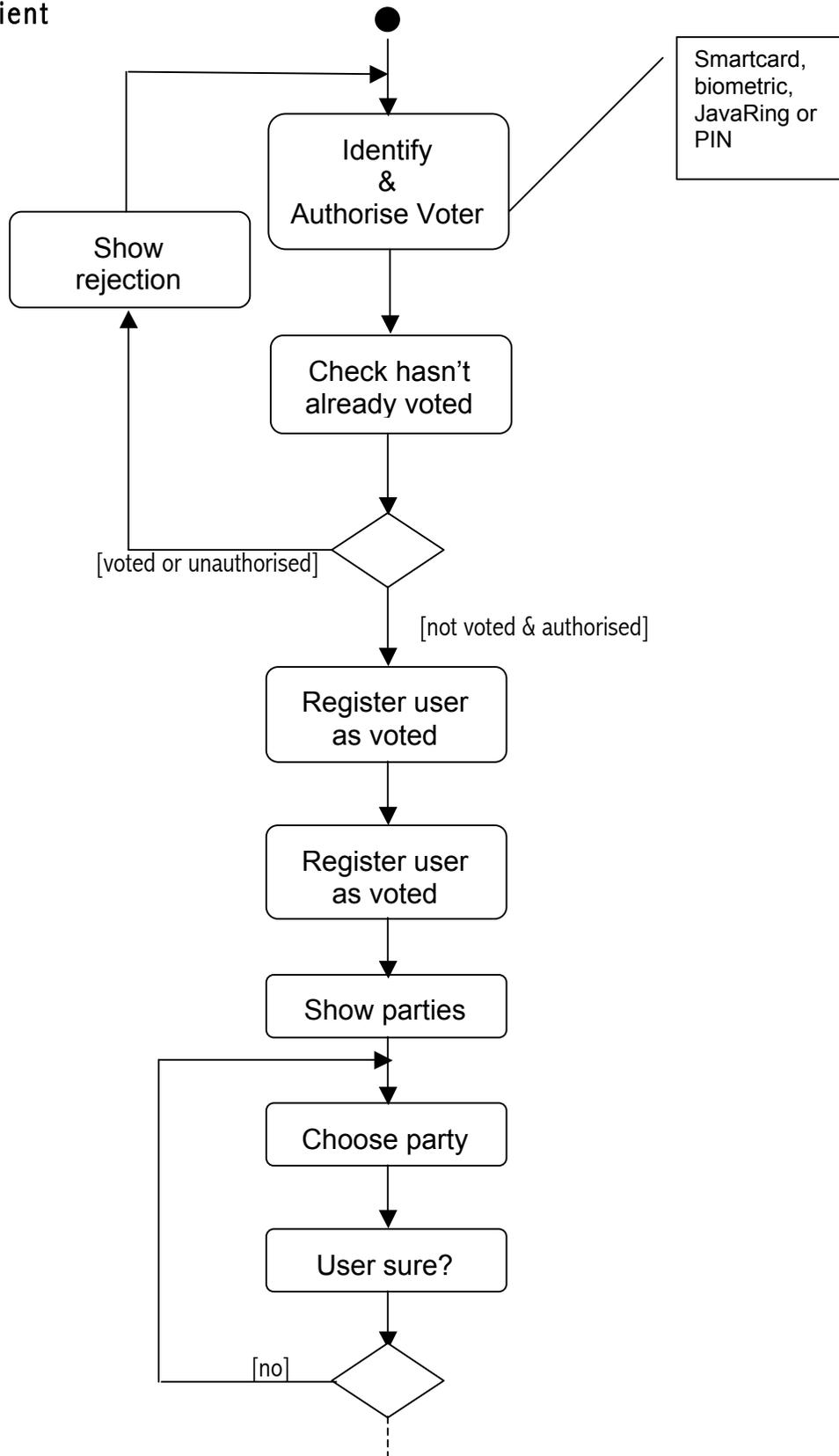Additionally packets are defined by 1's state which can take on one of the following states:
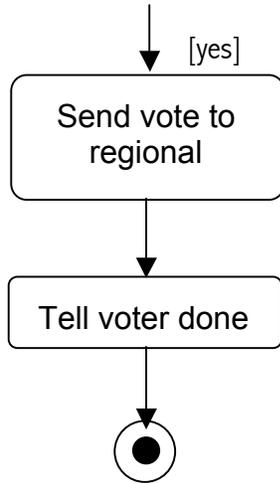
A   Authorisation check
C   Vote or total confirm
D   Diagnostic packet
E   Electoral roll check
T   Total for a party/choice
V   Vote packet
X   End communication

It is also worth noting that the MAC system described earlier is implemented within the **AuthSys** class.
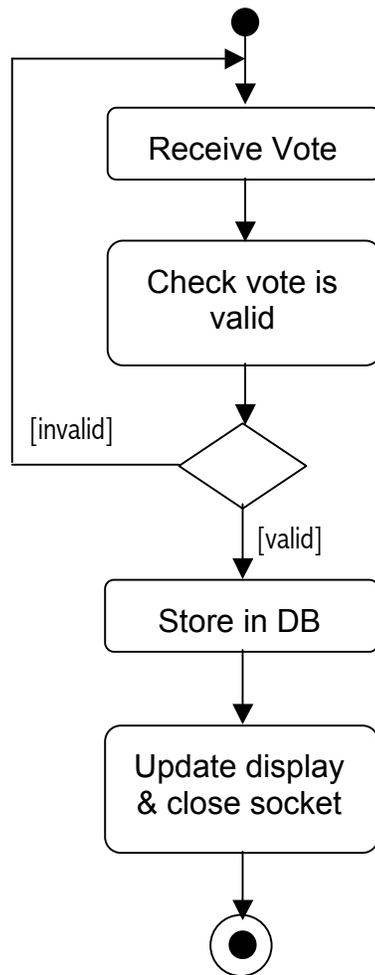
So from these class diagrams, the following activity diagrams can be derived:

FreeClient

```
                              ●
                              │
                              ▼
                    ┌──────────────────┐        ┌──────────────┐
                    │     Identify     │        │ Smartcard,   │
            ┌──────▶│        &         │───────▶│ biometric,   │
┌──────────┐│       │ Authorise Voter  │        │ JavaRing or  │
│   Show   ││       └──────────────────┘        │ PIN          │
│ rejection│◀──┐              │                 └──────────────┘
└──────────┘   │              ▼
               │    ┌──────────────────┐
               │    │   Check hasn't   │
               │    │  already voted   │
               │    └──────────────────┘
               │              │
               │              ▼
               │            ◇
 [voted or unauthorised]
                              │ [not voted & authorised]
                              ▼
                    ┌──────────────────┐
                    │   Register user  │
                    │     as voted     │
                    └──────────────────┘
                              │
                              ▼
                    ┌──────────────────┐
                    │   Register user  │
                    │     as voted     │
                    └──────────────────┘
                              │
                              ▼
                    ┌──────────────────┐
                    │   Show parties   │
                    └──────────────────┘
                              │
              ┌──────────────▶│
              │    ┌──────────────────┐
              │    │   Choose party   │
              │    └──────────────────┘
              │              │
              │              ▼
              │    ┌──────────────────┐
              │    │    User sure?    │
              │    └──────────────────┘
              │              │
              │   [no]       ▼
              └─────────────◇
                              ┊
```

[yes]

Send vote to
regional

Tell voter done

RTServer

Receive Vote

Check vote is
valid

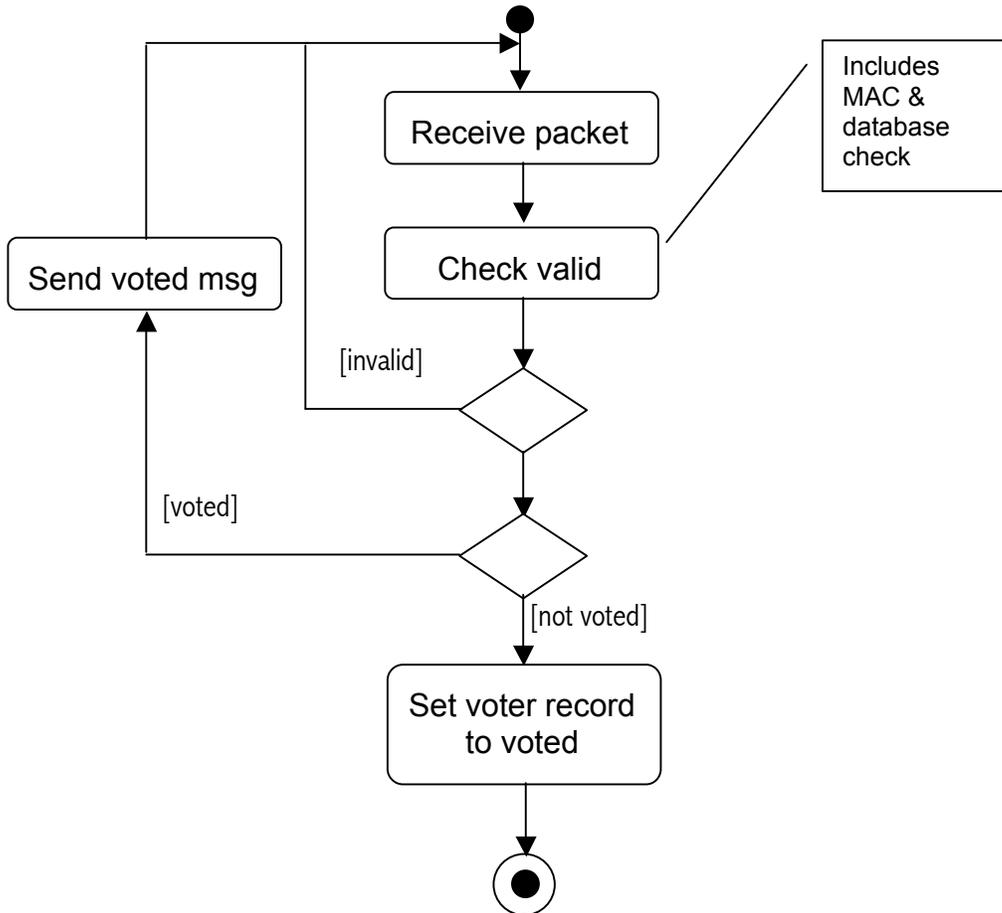[invalid]

[valid]

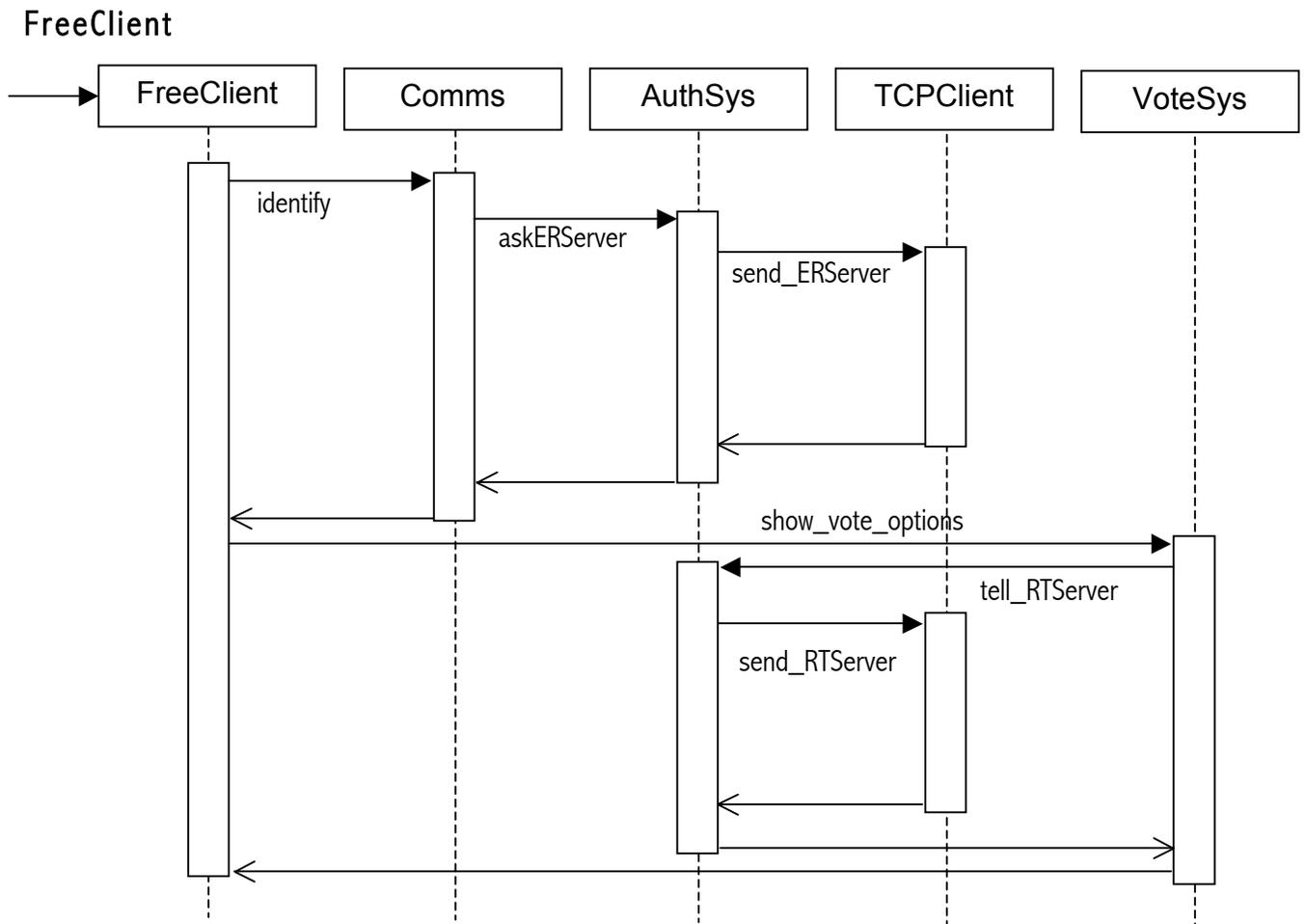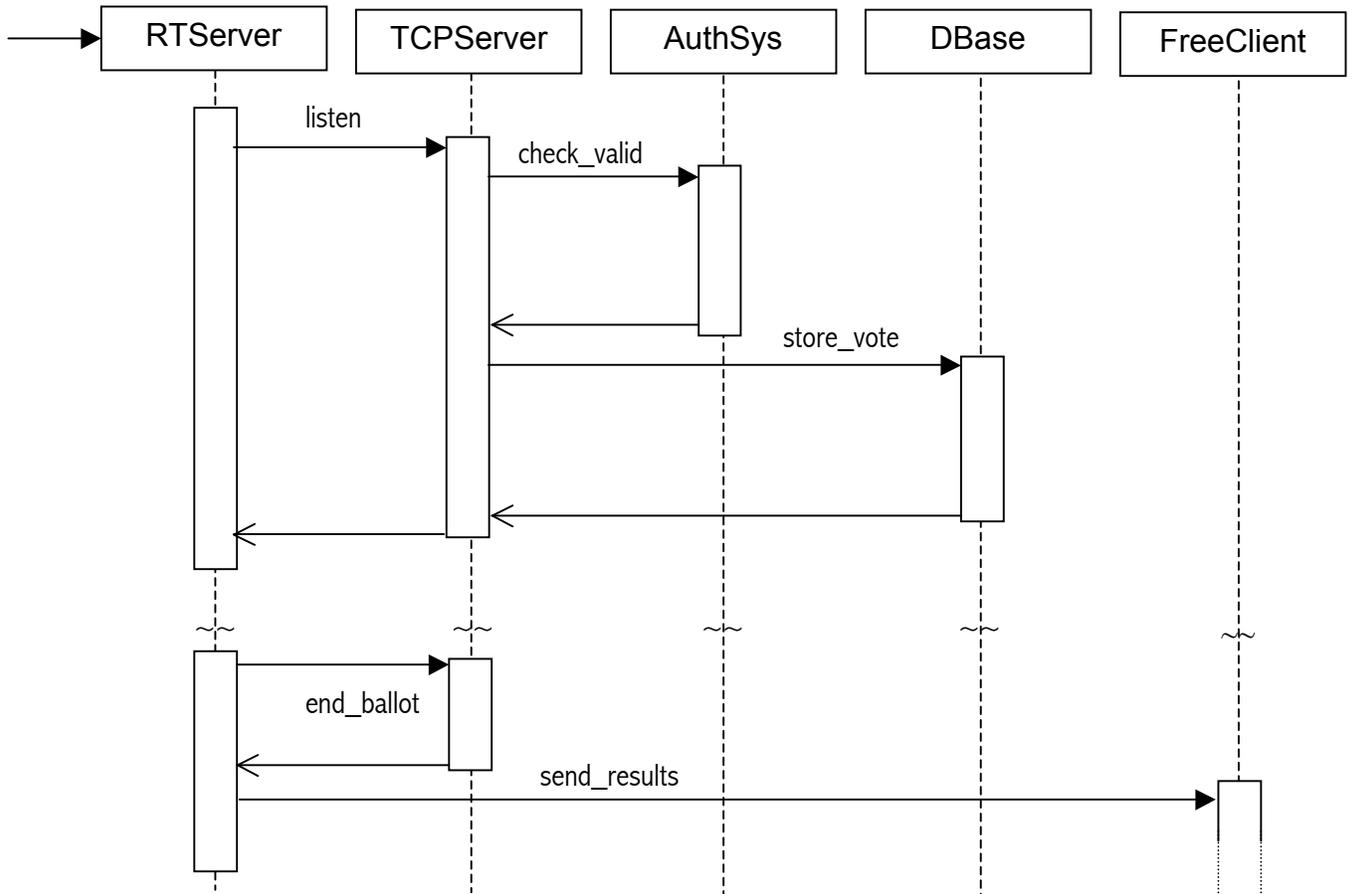Store in DB

Update display
& close socket

ERServer

From this analysis it became clear that while FreeClient would consist of mostly simple actions, it would nevertheless have in theory the most complex activity set of the three programs. However it must be noted that when RTServer is in regional mode it needs to adopt FreeClient's activity model at the close of the ballot so that it can send (in client mode) results to a totaller server. Thus this results in RTServer being, in essence, a hybrid of ERServer's and FreeClient's logic. However for design purposes this can be essentially ignored as the implication is merely that RTServer will need to run FreeClient code once it has finished its own server functions and thus no new logic is needed.

From these activity diagrams a series of schedule diagrams could be constructed to help understand the timing issues of the software. This also proved useful when, during implementation, it became apparent that the entire system would have to be threaded. Note that all interactions with GFX have been hidden to improve the clarity of the diagrams.

FreeClient

FreeClient	Comms	AuthSys	TCPClient	VoteSys

identify

askERServer

send_ERServer

show_vote_options

tell_RTServer

send_RTServer

RTServer

RTServer   TCPServer   AuthSys   DBase   FreeClient

listen
check_valid
store_vote
end_ballot
send_results
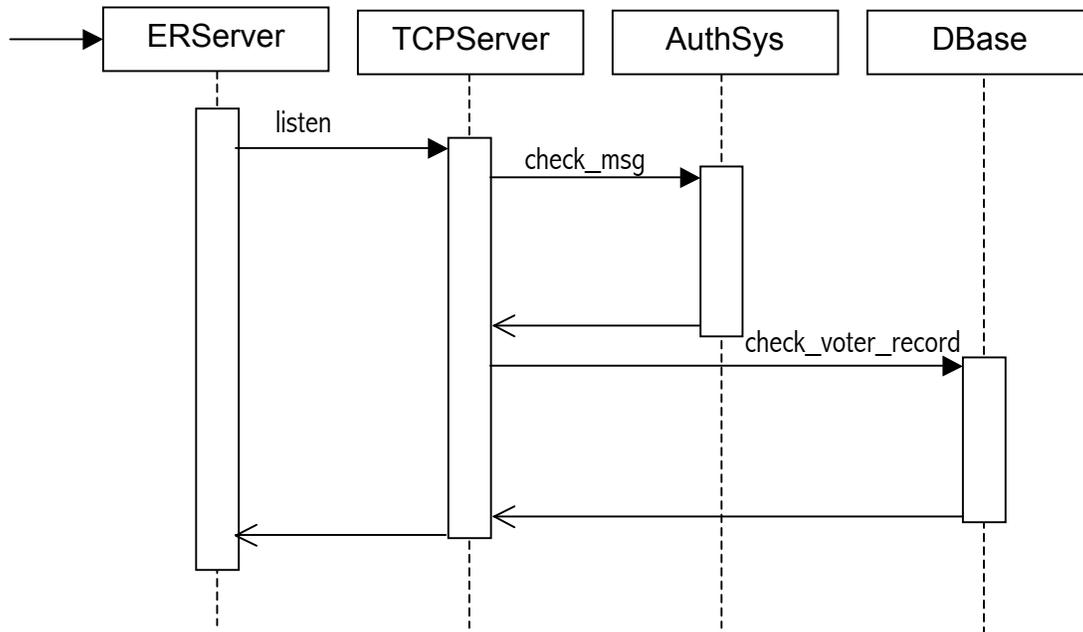
In this diagram one can see that after (x) repetitions of receiving a vote, the operator ends the ballot and then the FreeClient code is used to send the results to a totaller server.

ERServer

ERServer can be seen to be extremely similar to RTServer's functionality.

Design Problems Encountered

The Replay Attack

Commonly known as the replay or 'copy & paste' attack, it became apparent during design and analysis that a cracker could capture vote packets being sent by FreeClient to RTServer with a sniffer. The cracker would find a vote packet for their chosen party and then could send it to RTServer numerous times and thus throw the vote. RTServer would regard each packet as valid because they would not have been tampered, merely resent.

In any system one cannot guarantee security, but one must try to make it sufficiently infeasible (i.e. time-consuming & complicated) to discourage breaches. To address the replay attack I developed a time stamp system which, while not mathematically proven as secure, can be regarded as a good real-world solution:

- Send a vote packet with a time & date stamp of when it was sent, keeping a copy of the time & date stamp in the client's memory.
- On receipt of this packet, the server replies to the client, asking for the stamp again.
- The client sends its copy from memory and then removes it from the memory location.
- The server gets the copy, compares it with the time & date stamp on the original vote packet. If they match a vote is registered. If they don't it could be a fraud attempt!

Thus by removing the time & date stamp from the client's memory once it has been verified, any copies of the packet sent maliciously to the server will fail due to their stamp having already been 'used' during the valid packet's verification.

The only way to fool this would be to trap the vote packet, extract the time & date stamp from it and then intercept the server's packet asking for the stamp again. Then our hypothetical hacker would have to recreate the reply packet, with the correct MAC and reply. This process would involve IP spoofing, sniffing, incredible timing, some luck and access to the right passphrase and correct algorithms for the MAC. It is possible – but highly unlikely. And if it was done it would only get the hacker one extra vote, he'd need to repeat that process hundreds of times to have any impact on the election in progress.

Export Restrictions

The above problem could have been solved very simply with the use of encryption and digital signatures. However, as mentioned earlier, encryption systems are costly. Digital signatures incorporate encryption and also require some complex and generally tedious management techniques. Most importantly all these techniques are subject to export restrictions and so I developed the entirety of FREE without reliance on encryption, which was a testing but enjoyable intellectual challenge.

But, for those organisations able to import encryption technologies I decided to create an optional version of FREE which implemented SSL communications as an extra security layer – though keeping in mind its limitations as mentioned previously. This version would not use certificates and would be programmed with a free Java toolkit from Entrust.com, thereby keeping FREE cost-free. Thus export restrictions are navigated but those still concerned are given the extra option to use, if able, to have all FREE communications encrypted.

Biometric Scanning

One of the original design intentions of FREE was to implement biometric scanning to help improve on the authentication issues identified as problematic by the Californian Internet Voting Task Force and thus I entered into discussion with several manufacturers of fingerprint scanners. Fingerprint scanners were chosen due to their ease of use and so more probable user acceptance while also providing extremely high levels of security. Those companies that offered complete products tended to only support USB connections in the Windows9x environment – which was unacceptable with my commitment to Java. One company was more flexible, Veridicom, however after detailed discussion with their technical staff it became apparent that their programming APIs would not be compatible with Java.

So until biometrics become platform independent I developed a simpler password-based system which could be used within existing electoral roll structures and would offer adequate levels of security. However it is my goal to make FREE's user authentication design extensible so that any chosen security system can be used with minimum effort.

3. Implementation

Some technical details

FREE was programmed using Metrowerks' CodeWarrior Pro for Java 5 on MacOS. This provides an integrated development environment (IDE) including Sun's Java Debugging framework, Swing frame design and rapid application development (RAD) wizards to aid with the construction of graphical Java applications. CodeWarrior is widely recognised as a leading programming tool and was chosen due to its ease of use, availability on Mac, PC and Solaris as well as its excellent academic licensing programme.

By developing on the MacOS the developer is prevented from slipping into some bad habits which can easily become natural when developing in UNIX or Windows. 100% Pure Java software (i.e. programs which are totally platform independent) cannot rely on command line interfaces as seen in UNIX. MacOS doesn't have any command line features and as such forces good Java programming practices!

Additionally the MacOS provides some extremely easy-to-use networking features making testing FREE relatively trouble-free. An IP network was created between a PowerComputing PowerTowerPro Mac clone and an Apple iBook with standard twisted pair cable and a 3COM hub. With the free download of Apple's MacDNS software to create pseudo-domains it was simple to simulate internet connectivity between the various FREE applications.

Having committed to a cross-platform system which was free to use I had to find a database system with equivalent features. After some research on the Internet I found dtf/SQL by the German company sLAB which is free for non-commercial use and could be run on MacOS, Unix, Windows and OS/2. Unfortunately Java database connectivity (JDBC) would not be provided until release 2.00 which was already several weeks overdue. I waited but it became apparent that it would be an extremely long delay. Thus I went in search of a replacement and after considerable investigation I found Hypersonic SQL (HSQL), by Thomas Mueller. HSQL was in many respects an improvement on dtf/SQL in that it was a General Public License program like FREE thereby giving me full access to the source code. Also HSQL is interesting in that it is entirely written in Java which makes it as portable as FREE and extremely easy to integrate. The one negative aspect is that it does not support concurrent sessions but merely queues multiple queries.

Unsurprisingly profiling has shown that this does lead to performance deterioration as the number of concurrent FREE users rises. However because HSQL is extremely small and efficient the performance degradation is not prohibitively large. But it cannot be discounted that such a performance decline would indeed be a considerable concern for a system serving thousands of simultaneous connections. I would anticipate that organisations with the resources for such large servers would probably implement the trivial amount of JDBC code needed to make FREE use a different database that can handle concurrent queries. The other, more cost effective solution would be to simply use more RTServers and ERServers, thereby spreading the impact wider and decreasing the load on each database.

Also of interest is that every method which didn't override a Java method (because these must keep the same security settings as the method they override) had its security level set as `private` or `protected.` Private methods and variables have the highest level of security as they can only be accessed within their own class or method, respectively. Protected methods and variables are less secure

but they can only be accessed from within their own package. This adds a level of security within Java's own framework – making it harder, though not impossible, for a malicious class to be inserted.

Problems & Bugs Encountered

To Applet or not to Applet?

It had been my intention to make FreeClient a Java applet. This would enable simple program distribution through the standard browser interface and would also allow use of SSL for those who wished it without any additional code.

However a little known property of applets reared its ugly head early on in the coding process. Java's security model prevents applets from accessing servers other than the one which delivered them unless authenticated with complex and expensive certificate based technology. There is solid reasoning behind this, however my previously asserted commitment to avoid the costs of certificate technologies meant that FreeClient would have to be an application otherwise the privacy features of separate ERServer and RTServer would be rendered inoperable.

While this decision protected the privacy status of FREE it has made distribution of the FreeClient more arduous. I haven't found a simple solution to this, however when FREE is launched to the open source programming community this is one of the first issues I would like to address.

The Java Sockets Problem

During the process of coding the network communications classes of **TCPServer** and **TCPClient** it became apparent during testing that there was tremendous inconsistency in the reliability of FREE's network connections. Additionally quite often a communications session would never successfully end.

As this bug was further explored it became apparent that this problem was not consistent between MacOS and Solaris (the two testing platforms). In fact on Solaris FREE's networking performance was far more consistent and reliable. Concerned by this unacceptable state of affairs from a system committed to being cross platform I decided to temporarily replace the classes with equivalent code from Sun's Java Tutorial. This had no positive effect and in fact seemed even less reliable!

This bug was halting further progress with FREE as testing any new features became impossible if the programs wouldn't communicate. Thus an intensive research effort was begun, e-mails were sent to Dr Mike Joy (who passed it on to his associates) and web sites such as java.sun.com, gamelan.com and metrowerks.com were trawled.

A sort of answer was finally found on Apple's Developer Connection Java knowledge base. An article explained how Java's networking object – sockets – use output and input streams which were originally created with file access in mind. Because of this, these classes use the end of line (EOL) character of the operating system the program is currently running on so that they can correctly interpret local files. However MacOS, Windows and Solaris/UNIX all use different EOL characters and due to the Internet's heritage only Solaris/UNIX uses the same character as is used within TCP/IP rendering Windows and MacOS networking unpredictable. Thus the article explained that for Java networking programs to be fully platform

independent the command `println` should be avoided due to its automatic insertion of the operating system's EOL character and the programmer should insert the correct symbol within their code using `print`.

Unfortunately this didn't solve the problem. So the **TCPServer** and **TCPClient** code was rewritten to extract the processing of packet data into protocol classes. This made the code cleaner and left only purely networking logic in **TCPServer** and **TCPClient**. While in the long run this significantly eased development it didn't solve the problem in hand which was beginning to threaten the overall schedule as specified in the project specification. In fact in my progress report I had to indicate some slippage due to the delays being caused.

Dr Joy and his students had, excuse the pun, no joy in solving the problem either. In fact I cannot explain why the eventual solution works. I was racking my brains and just decided to use `println` <u>and</u> manually insert the TCP/IP EOL character and this rendered FREE's networking reliable and fast!

This bug had become so tiresome and troublesome purely because it was extremely hard to track. It was clear from diagnostics on the hub that the correct data was being sent and that the correct software was receiving the packets. The only symptom was a consistent failure for connections to close and so obviously working from this resulted in the lengthy process of identifying its source.

Threading

Before the previous bug became apparent the patchy networking was seen to be unresponsive. Additionally it seemed that the GUI was freezing with alarming regularity. With a little further exploration it became clear that this only occurred during network operations.

Of course it was then obvious that due to network latencies the system would be held up. The answer was to thread FREE software into two portions, networking and the rest, thereby keeping the system responsive. It was fortunate that FREE's design enabled threading to be easily implemented once I taught myself how it was done!

It must be noted that TCPServer was always threaded, each client connection is spun out into a new thread leaving the server free to listen for new connections.

The Message Authentication Code

While implementing **AuthSys** for use across FREE I encountered a bizarre problem. The same MAC algorithm appeared to not always create the same answer when provided with the same input. Had I misunderstood what message digests and MACs were all about? I re-read articles and books on the matter, I had understood what they did. So what was going on?

I would send a packet with a MAC and then on the receiving end recalculate the MAC from the packet's data portion — yet it wouldn't ever match the MAC in the packet. Was my network corrupting the packets? I dissected network traffic and found it wasn't. It was time to once again trawl the Internet and on an obscure web site holding old lecture notes I found some work on message digests. Message digests output a byte stream and I was using the standard `toString()` method before adding the MAC to a packet.

However it became clear from these notes that the `toString()` method doesn't actually work as expected. The MAC needed conversion to hexadecimal before being converted to a string. Working off the example code I adjusted **AuthSys** and almost instantly my MACs were being validated as ok. Problem solved.

HSQL's deprecated method

All of the classes for HSQL are stored in a jar file (the standard compressed format for keeping Java packages together) but the source code is also supplied separately. Database implementation was left to the end of implementation partly due to the delay in finding a system but mainly for testing purposes. All errors within FREE could be isolated before introducing the unknown quantity of a new database system.

So having completed testing, HSQL was implemented. But every time a new database was created in the FREE initialisation code an extremely long exception was thrown. This exception listed methods which went from deep inside FREE, through the HSQL code and into Java's own code. After dissecting the exception I located the problem in the HSQL code. A call to **java.util.Properties.store()** was bringing up the exception, claiming that in effect the method didn't exist. I checked the Java 1.2 API documentation to confirm that it did indeed exist. This was a key bit of code for HSQL as it allowed the system to create persistent data stores so a solution had to be found, I couldn't just cut out that bit of code.

I thought perhaps it was a bug with my implementation of the Java Virtual Machine so I upgraded the MacOS Runtime for Java (MRJ) to the latest version but this had no impact. Perhaps it was an outstanding bug that still hadn't been fixed so I checked Apple's technical support website. It was here they I finally understood the problem... I wasn't running Java 1.2, I was using Java 1.1.8 but HSQL had been compiled for Java 1.2. Fortunately HSQL had an automated system for translating it to Java 1.1.x code but it only worked in a Windows environment. So I transferred the code and, after a few false starts, managed to compile a new jar file that was Java 1.1.x compliant and so didn't use the offending **java.util.Properties.store()** method. This was transferred back to the development system and HSQL worked perfectly. Problem solved, but if I hadn't had access to HSQL's source code it would have taken me much more than a couple of hours to resolve the issue. This is an indicator of the effectiveness of open source development.

Testing the systems

An incremental approach was taken so that as every new unit of functionality was implemented it was tested instantaneously. This immediately narrows down the complexity of the analytical task. By doing testing during coding one also considerably improves the test coverage over the lifecycle of the whole project. While I don't have the tools to calculate a qualitative value for the test coverage I would argue that it is undoubtedly a high value: There are currently not more then several thousand lines of code in FREE which were not only tested when implemented but also were subjected to a testing regime at the end of the development process.

To ease this task a suite of testing routines was created in the form of the FREE Testing Suite (FTS), a Java program with GUI access to launching tests and reviewing their results. This tool also provided simple profiling of system performance. With this tool all defensive programming (which was used throughout the FREE system) could be checked by the automated delivery of unlikely or 'impossible' packets. Additionally

the implementation of exceptions could be tested to ensure that they were thrown up to the correct class and that when things did go wrong connections were closed and objects were cleaned up gracefully.

FTS was also used in boundary testing by sending packets of no or small size along with huge packets. Problems were highlighted, especially with smaller packets — while fixing the code related to this area it became apparent that a large performance boost could be made by pre-checking vote packets prior to sending them through the full authentication code. Due to their specification vote packets cannot be less than 63 characters and so any packets headed with a 'V' and less than 63 characters in size can thus be immediately discarded. The processing time saved by not passing these packets through the MAC routines was highlighted when the code was rechecked with the profiler: RTServer was now taking 5-10 ms on under-sized packets compared to a previous 300-500 ms.

I was also concerned about buffer overflow situations with malicious crackers sending massive packets to FREE to crash the servers. The packet data structure is based on Java's **String** data type, however in testing by a fellow CBS student, Neil Ferguson, a single **String** variable stored the entire text of the Old Testament — all of 3Mb! Even so, any attack consisting of such large packets would probably stumble in routers well before it reached a server, leaving me assured that this would not be FREE's Achilles Heel.

Another part of FTS is the ability to perform stress tests by simulating multiple clients talking to the server simultaneously. I felt this to be a key to proving that FREE is a scalable solution as it was initially designed to be. Firstly I was pleased that ERServer performed excellently in tests, though admittedly it has the simplest task of the three FREE programs. RTServer fared significantly less well: Any test with over 5 simulated clients would result in failed vote authentication, providing a major blow to claims of scalability. However the bug was not consistent and not entirely repeatable with sometimes more votes than usual succeeding and sometimes none at all.

It took a considerable effort to track down this problem, but it was eventually found. My first approach was to use the Java debugger (JDB) however this met with unmitigated failure. The JDB communicates with the software being debugged through TCP/IP but as all of FREE's communications are also in TCP/IP the whole operation was slowed down to an unusable pace. Additionally whenever one tried to access more detailed debugging information FTS data couldn't get through to RTServer, creating new errors.
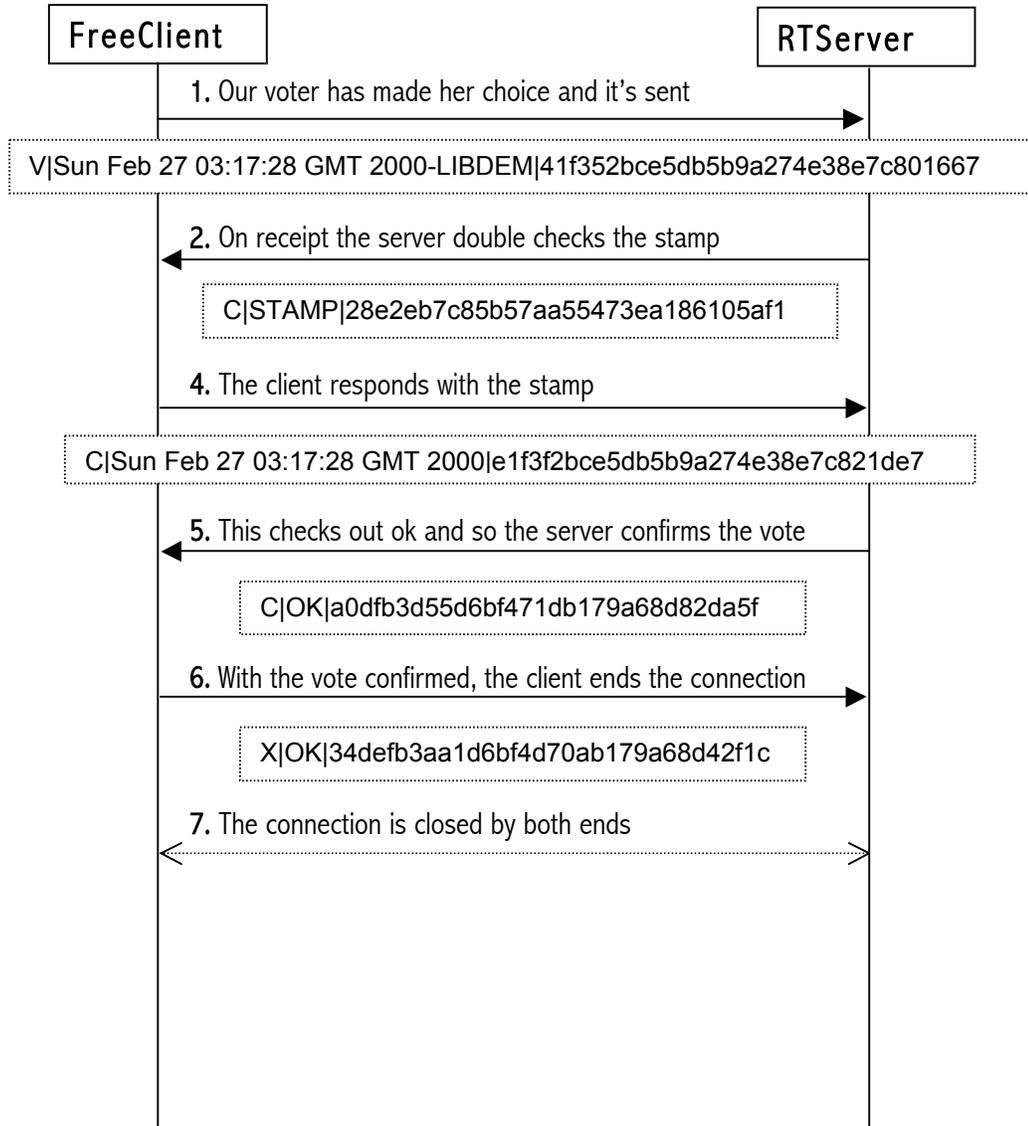
Thus I worked on adding my own debugging read outs and adding excessively fine-grained exception handling which could later be revoked. This, after a long iterative process, directed me towards the data structures used to hold the vote security stamps while RTServer waits for confirmation from the client with a second copy. However while I knew that the problem was related to this code, I had no indication of what the nebulous problem actually was. All the exception-related data provided was simply 'null'. So I embarked on some thinking, reading and research. After some time it became apparent that due to the multi-threaded nature of RTServer there could be some race conditions occurring when different connections tried to access and alter these data structures. Being new to thread concepts I hadn't considered these issues, however thanks to testing I resolved the issue by changing the data structures to **java.util.Vector** which has built in synchronisation and is thus 'thread-safe'. RTServer now successfully completes the FTS tests.

Finally it is worth noting that by releasing FREE with its source code under the General Public License across the Internet it is likely to come under the scrutiny of a large number of users in a wide variety of

different settings. The end result is access to a large number of capable testers who are likely to spot previously unseen problems, and vitally they are also able to fix these problems themselves without any delay – a resource any corporate development program cannot match.

Graphic Summary

Perhaps the best way to sum up what FREE does and how it works is to present a 'day in the life of a vote' in a graphical format. We pick up the story from where a valid voter has been authorised and is just mulling over who to vote for… (the dashed boxes indicate the actual packets sent)

```
┌─────────────┐                              ┌─────────────┐
│  FreeClient │                              │  RTServer   │
└─────────────┘                              └─────────────┘
       │  1. Our voter has made her choice and it's sent   │
       │ ────────────────────────────────────────────────▶│
 ┌─────┴───────────────────────────────────────────────────┴─────┐
 │ V|Sun Feb 27 03:17:28 GMT 2000-LIBDEM|41f352bce5db5b9a274e38e7c801667 │
 └───────────────────────────────────────────────────────────────┘
       │  2. On receipt the server double checks the stamp │
       │◀────────────────────────────────────────────────  │
       │   ┌───────────────────────────────────────────┐   │
       │   │ C|STAMP|28e2eb7c85b57aa55473ea186105af1   │   │
       │   └───────────────────────────────────────────┘   │
       │  4. The client responds with the stamp            │
       │ ────────────────────────────────────────────────▶ │
 ┌─────┴───────────────────────────────────────────────────┴─────┐
 │ C|Sun Feb 27 03:17:28 GMT 2000|e1f3f2bce5db5b9a274e38e7c821de7  │
 └───────────────────────────────────────────────────────────────┘
       │  5. This checks out ok and so the server confirms the vote │
       │◀────────────────────────────────────────────────  │
       │   ┌───────────────────────────────────────────┐   │
       │   │ C|OK|a0dfb3d55d6bf471db179a68d82da5f       │   │
       │   └───────────────────────────────────────────┘   │
       │  6. With the vote confirmed, the client ends the connection │
       │ ────────────────────────────────────────────────▶ │
       │   ┌───────────────────────────────────────────┐   │
       │   │ X|OK|34defb3aa1d6bf4d70ab179a68d42f1c      │   │
       │   └───────────────────────────────────────────┘   │
       │  7. The connection is closed by both ends         │
       │◀ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ▶ │
       │                                                   │
```

The Web Site

When thinking about how to provide detailed documentation to accompany the release of FREE it soon became clear that HTML would be the best format in which to provide structured and totally cross platform access to all the necessary information. Additionally this ties in nicely with Java's own JavaDoc automated documentation system whose output format is also HTML. JavaDoc's output is being used to provide the specific implementation details useful to programmers wanting to extend FREE's source code.

These HTML files will be provided with every distribution of FREE, but additionally they will be made available from the FREE web site at http://www.thecouch.org/free/ so that updates and changes can be easily made available to all users. This web site also provides a central point for co-ordinating the development of FREE's future versions as well as repository for code upload and download, FREE news and a access to the associated research including the *People Power* essay.

Thus if the reader is interested in obtaining a copy of FREE or would like to examine to documentation please visit http://www.thecouch.org/free/

4. Self Assessment

What is the technical contribution of this project?

FREE implements novel techniques based, in part, on the theoretical work of David Chaum. As far as I can tell this is the first time a non-commercial project has produced a feasible platform-independent system that can conduct ballots on local to national scales. In fact FREE offers, in my opinion, superior features to all commercial systems reviewed. This is partly thanks to its use of Java but also due to my commitment to making the system useable without any reliance on technologies not yet widely available and/or affordable.

Additionally by making FREE available as an open source program the whole Internet community can freely use and modify the source code. Hopefully this will jump start an effective non-commercial and independent system for conducting ballots. The desired outcome being that commercial organisations become sidelined from what I believe to be an area that must be kept non-partisan and non-profit.

Why should this contribution be considered either relevant or important to computer & business studies?

The *People Power* essay highlights the growing influence ordinary people have on the decisions governments and corporations make. We all come in contact with these organisations by either working for them or using their services; thus we must come to understand and react to these changes. One of the key results of the oncoming change is being clearly manifested as a powerful desire for greater openness and accountability.

These are the central values of democracy but democracy itself is being seen to wane and lose its citizens' interest. I believe the Information Revolution provides us with tools to not only return democracy to where it once was, but to strengthen and improve it beyond our current expectations.

The first step in doing this is making the current democratic processes more flexible and, as voting is the core activity of democracy, I have worked on it by creating FREE.

Computer and Business Studies (CBS) should really be called Information Revolution Studies: Computer technology and Business practices are more widely seen and felt than ever before. Governments and charities are using business management techniques while everything from schooling to Hoover manufacture is being influenced by computer technology. CBS looks at these issues by teaching us how to study, create and use the complex socio-technical hybrids we find in modern organisations in holistic way.

Despite what many might argue companies cannot exist in a vacuum working solely for profit. They must look at environmental, social and economic conditions together — a failure to do so is bad for business, as diverse companies ranging from Shell to the Body Shop now recognise.

This projects highlights the importance of non-governmental organisations (NGOs) and students of CBS must learn to tap NGOs so that they can better understand those all important environmental, social and economic conditions.

Finally, by reinforcing the democratic process with electronic voting, perhaps other stakeholders can be reminded that they cannot simply plough past the will of citizens for their own ends. Democracy can work and will continue to do so.

How can others make use of the work in this project?

The code can, of course, be used to provide electronic voting systems! The concepts and some of the code could also be used to provide other services — such as market research systems — which also require privacy concerns to be addressed.

I hope that by creating this fully working proof-of-concept system as open source cross-platform software an example will be set which encourages others to provide open source systems that work to nourish and rejuvenate democracies.

The *People Power* essay provides a framework for examining particular cases of InfoRev technology impacting the political process and NGOs in more detail. It also offers an extensive review of the current state of development in terms of e-democracy which may help other researchers to work in the most productive directions. In particular I feel that there is a large amount of work to be done on electronic communities to help build on earlier positive, but short-lived, projects.

Why should this project be considered an achievement?

I have further explored my concept of the Revolution in Civilian Affairs which was first proposed in *Information Age War: The changing nature of conflict within the context of the Information Revolution* (1999). From this I explored concepts such as the Organisational Revolution, one of the least recognised portions of the InfoRev. Subsequently some unique typologies were created along with novel analysis of the convergence of technology, the growing influence of NGOs and the current malaise in Western democracies. This work was not just a literature review but offers new thinking on the area and provides interesting new directions for others to study. For example, a key area for public policy researchers should now be on how

to leverage NGOs' increasing use of technology so that ideas can be provided to improve NGO involvement in the governmental decision making-process.

The FREE electronic voting system is also an achievement: It provides private, secure and scaleable voting over the Internet at no cost and without the need for rare or expensive technologies. Additionally it is freely available to anyone with an Internet connection at no cost and with no strings attached. This is completely novel as all other systems are either commercial, prototypical or reliant on exotic technologies being in the hands of all voters.

What are the weaknesses of this project?

I feel that the People Power essay could have had more research done and perhaps a couple of detailed case studies should have been provided to better explain just how things have changed. Additionally I would have liked to spend some more time on the role media play in politics and how this might be changing. However it did become apparent that this area is a huge work in itself.

FREE suffered from my previous lack of knowledge with regards to Java's threading and networking features, its design and implementation might have somewhat differed if I had held that detailed knowledge previously. Additionally, given more time I would have liked to try and find a biometric scanning system that could be used with Java to provide an example voter authorisation implementation.

Despite these misgivings I still feel justified in being proud of the achievements made with this project both in terms of critical analysis and technical developments.

5. Conclusions

During the development of this project its objectives were validated by a massive flurry of commercial e-democracy activity along with a raft of papers in journals on the area, including *Communications of the ACM*. Additionally I have encountered a rapidly increasing number of examples that illustrate the growing use of technology in the political process which clearly justifies my work in *People Power*. My one regret about the full title of the project is that some people find its verbosity off-putting, I continue to search for a clear way to convey the concepts I have explored.

This was an ambitious project, though the implementation of biometric voter authorisation might have been stretching it too far. However overall I feel it to have been a great success, I have explored above how I feel it to be a valid contribution to my field and in the process I have learnt inordinate amounts.

This work has also raised a large number of questions: Will citizens accept electronic voting? Can a non-commercial electronic system succeed? Can we ever trust electronic systems? Will e-democracy ever have a measurable impact? Is the digitisation of media playing a role in the political process? Is the Internet really that democratic? And so on… this is an exciting area to study with new developments constantly being announced. My next steps in this area will be to administer FREE's open source development and to write a research paper examining just how different Internet media is to the old mass media?

The technical development took a pragmatic approach, reviewing the current theoretical cutting edge and then using these concepts in a real world context that didn't rely on tomorrow's developments. This has its disadvantages – the system cannot not be mathematically proved as secure and private, but I strongly feel that the area needed a kick start with development today.

Difficulties were encountered, partly as a result of transposing theoretical systems to reality but mostly due to my own lack of technical knowledge. The key lesson learnt from this project is undoubtedly that even more technical research than I anticipated is needed before design even be contemplated.

Java proved to be a good choice, it is a wonderfully robust language to use and offers so many useful pre-made objects. However its debugging and testing infrastructures leave a lot to be desired. An excellent documentation architecture is provided in Java with JavaDoc, now the same is needed for testing. Additionally I must strongly question the designers' choice of using TCP/IP to communicate debugging information. But overall Java's clear syntax, network awareness and cross-platform capabilities win me over.

I was extremely pleased with the testing phase which successfully flagged up some important problems. Additionally my detailed design and analysis diagrams helped me to address these issues by clarifying my understanding of class interactions and timings.

The biggest test for FREE will be when it is released to the public via the Internet, only when the extremely picky open source programmers run their fingers through the code and the e-mails come bouncing back will I truly know whether FREE has been a success.

Nevertheless as a result of this project I have produced: a significant work in the *People Power* essay, the FREE system consisting of three key programs and a testing suite, a web site which includes the software documentation, a presentation, specifications, timetables, a progress report and the document you hold in your hand.

Why? To better understand the phenomenon that lets ordinary people make a difference: People Power.

Acknowledgements

6. Bibliography

Anderson, K *BBC News Online* Opening up the digital democracy 10th January 2000 http://news.bbc.co.uk [And00]

Anonymous *Maximum Security* (2nd Edition) Sams Publishing 1998

apc.org *APC Programmes* Real Life Strategic Uses of APC Networks 1999 http://www.apc.org

Arquilla, J & Ronfeldt, D *In Athena's Camp* RAND 1997

Asimov, I *Robot Dreams* Vista 1997

Barmé, G & Ye, S *Wired 5.06* The Great Firewall of China pp138 June 1997

Bastard: Global Edition Kosov@ Special May 1999

Benaloh, J & Tuinstra, D *Receipt-Free Secret-Ballot Elections (Extended Abstract)* Clarkson University

Benaloh, J & Yung, M *Distributing the Power of a Government to Enhance the Privacy of Voters (Extended Abstract)* Yale University & Columbia University

Benaloh, J *Verifiable Secret-Ballot Elections* Yale University 1996 [Ben96]

Bender, W et al *IBM Systems Journal* Enriching communities: Harbingers of news in the future Vol. 35, NOS 3&4 1996

Borger, J *The Guardian* Internet campaigns mark new era in political advertising 8th January 2000

Brand, S *The Media Lab* Penguin 1988

Brockman, J *Digerati* Orion Business 1996

Campione, M & Walrath, K *The Java Tutorial Second Edition* Addison-Wesley 1998

Castells, M *End of Millennium* Blackwell 1998

Castells, M *The Power of Identity* Blackwell 1997

Castells, M *The Rise of the Network Society* Blackwell 1996

Chaum, D *Communications of the ACM* Security without Identification: Transaction Systems to make big brother obsolete October 1985 [Chaum85]

Chomsky, N & Herman, E *Manufacturing Consent: The Political Economy of the Mass Media* Pantheon Books, New York

Cohen (Benaloh), J *Improving Privacy in Cryptographic Elections* April 26, 1994

Computing Bulletin 'Voting in UK elections may be allowed by telephone' 3rd June 1999

Dyson, E *Release 2.0* Broadway 1997

Dutton, W & Elberse, A & Hale, M 'A Case Study of a Netizen's Guide to Elections' *Communications of the ACM* pp49 December 1999

EBallot *Newscenter* DELL survey: 78% want to vote online in major elections July 1999 http://www.eballot.net

Eckstein, R & Loy, M & Wood, D *Java Swing* O'Reilly 1998

Eriksson, H & Penker, M *UML Toolkit* John Wiley & Sons, 1998

Fast Company December 1999 pp304

Flannagan, D *Java in Nutshell* (2nd Edition) O'Reilly 1997

Garfinkel, S with Spafford, G *Web Security & Commerce* O'Reilly 1997

Gilly, D et al *UNIX in a Nutshell* O'Reilly 1992

Hafner, K & Lyon, M *Where Wizards Stay Up Late* Touchstone 1996

Hagen, M *A Typology of Electronic Democracy* 1996 (Heavily abridged English translation of German original)

Jonscher, C *Wired Life* Bantam 1999

Kelly, K *Out of Control* Perseus 1994

Kernighan, B & Pike, R *The Practice of Programming* Addison-Wesley 1999

Kitcat, J *Digital Living: The Socio-Economic Impact of the Internet* 1999 (unpublished)

Kitcat, J *Information Age War: The changing nature of conflict within the context of the Information Revolution* 1999 (unpublished)

Larsen, K 'Voting Technology Implementation' *Communications of the ACM* pp55 December 1999 [Lar99]

Leigh, D 'Global Disclosure' *The Guardian G2* pp8 31st January 2000

Leslie, J *Wired 7.11* Operation Phnom.com pp230 November 1999

McGeary, J *TIME* 'Mohandas Gandhi (1869-1948)' December 31 1999

McLuhan, M & Fiore, Q *The Medium is the Massage* 1967, remastered edition HardWired 1996

Meeks, B *Communications of the ACM* Better Democracy Through Technology pp75 February 1997

Meeks, B *Communications of the ACM* Dragging a Kicking-and-Screaming Government into the 21st Century pp13 September 1996

Millennium Whole Earth Catalog Political Tools: Electronic Democracy 29-10-1999

Mullen, R *The HTML 4 Programmer's Reference* Ventana 1998

Negroponte, N *Being Digital* Hodder & Stoughton 1995

Oaks, S *Java Security* O'Reilly 1998

Oaks, S & Wong, H *Java Threads* (2nd Edition) O'Reilly 1999

Plant, S *Zeros + Ones* Fourth Estate 1998

Preece, J et al *Human-Computer Interaction* Addison-Wesley 1994

Rochlin, G *Trapped In the Net* Princeton University Press 1997

Rosenfeld, L & Morville, P *Information Architecture for the World Wide Web* O'Reilly 1998

Schwartz, J *The Art of the Long View* Currency Doubleday 1996

Special Edition *Mute* Tactical Media Issue 13

Standage, T *The Victorian Internet* Phoenix 1998

Stoll, C *The Cuckoo's Egg* Pocket Books 1990

Stoll, S *Silicon Snake Oil* Anchor 1995

Symmes, P *Wired 6.02* Che Is Dead pp140 February 1998

Toffler, A & Toffler, H *PowerShift* Bantam 1990

Toffler, A & Toffler, H *The Third Wave* Banatam 1981

Turkle, S *Life on the Screen* Phoenix 1995

Various *The Times Interface* Online Democracy special pp10-11 22nd September 1999

Various *Wired 8.01* Wired Diaries pp84 January 2000

Wang, W *Steal this Computer Book* No Starch Press 1999

Watson, R & Akselsen, S & Evjemo, B & Aarsæther, N 'Teledemocracy in Local Government' *Communications of the ACM* pp58 December 1999

Young, H *The Guardian* The Internet will take over politics in good or bad ways 6th January 2000

Endnotes

[i] Source: K Larsen Voting Technology Implementation *Communications of the ACM* December 1999

[ii] These companies are rather secretive and so only sketchy details are emerging of what they *actually do*!

[iii] Sources: J.C.Benaloh *Verifiable Secret-Ballot Elections* 1996, *Receipt-Free Secret Ballot Elecitons*, *Improving Privacy in Cryptographic Elections* 1994, *Distributing the Power of a Government to Enhance the Privacy of Voters*.

[iv] These systems have been pioneered by Akamai and Sandpiper, two companies with meteoric IPOs and already have a huge roster of customers including Apple, CNN, Disney etc who are keen to optimise their customer's Internet experience.

[v] Source: J.C. Benaloh *Verifiable Secret-Ballot Elections* 1996 pp113

[vi] Source: D Chaum *Communications of the ACM* Security without Identification: Transaction Systems to make Big Brother Obsolete October 1985 Volume 28 Number 10

[vii] Source: K Anderson *BBC News Online* Opening up the digital democracy 10th January 2000

[viii] For those unsure of what a message digest is: It is a digital fingerprint of some data. Data is passed through an algorithm that creates a sequence of bytes from the data stream. This digest (or hash) cannot tell you how much data or what data there is, but it is related in some irreversible way. The key point is that a message digest algorithm must provide assurance that it is computationally infeasible to find two messages that produce the same digest thereby preventing the original data from being substituted undetected.