Developing Requirements for Electronic Poll Book Systems: the Distributed Systems View and Challenges

Matthew Desmarais Laurent Michel Alexander Russell Alexander Shvartsman

Center for Voting Technology Research
Department of Computer Science & Engineering
University of Connecticut
371 Fairfield Way, Unit 2155, Storrs, CT 06268, USA

Abstract

Electronic poll books are computerized systems that replace paper-based voter lists, having the potential for speeding up Election Day check-in at the polling place, and making voter history records and voter lists more accurate by reducing human errors in dealing with printed voter lists and post-election transcription. At the same time, electronic poll books are non-trivial distributed computing systems and ensuring correctness, security, integrity, fault-tolerance, and performance of such systems is a challenging engineering problem. This paper deals exclusively with the distributed system aspects of electronic poll book solutions and focuses on the obstacles that are inherent in any distributed system that must deal with failure and asynchrony while providing a consistent and dependable service. We review several requirements that need to be satisfied by electronic poll book systems, then we discuss selected important results from distributed computing research that the developers of electronic poll book systems need to be aware of. The bottom line is that electronic poll book development is an attractive application domain for the research results in dependable distributed computing.

National Conference on State Certification Testing of Voting Systems, May 19-20, 2015, Seattle, WA, USA

1 Introduction

Voting systems are an integral component of the modern electoral procedures and an essential part of our democratic process. Such systems are composed of several entities working in concert: e.g., "Voter Registration Systems", "Election Management Systems", "Voting Terminals", and "Poll Book Solutions". The first of these has been computerized for some time using well-understood database technologies. The HAVA act of 2002[1] mandated a nationwide modernization of the voting infrastructure and led to the broad adoption of computerized solutions for the next two components. Consequently, all 50 States initiated efforts to select suitable digital voting solutions. The private sector rushed to market with a number of hastily created and often inadequately engineered solutions. The net result was an adoption of products that suffered from severe flaws at multiple levels: poor engineering, lack of resilience against the most elementary tampering or simple mis-configuration, the illusion of safety from misuse of cryptography, and—in general—a under-appreciation for the complexity of the electronic election systems that in essence need to be built as sophisticated distributed systems.

On the State of Technology. Dislocation of theory and practice in computing can lead to significant (and, in some settings, newsworthy) problems with basic computing infrastructure. Indeed, a particularly striking example arises in the context of electronic voting technology. The UConn Center for Voting Technology Research—directed by the authors and closely working with Connecticut State government—benefits from exposure to the nation-wide problems in electronic election systems created by a stark disconnect between theory and practice. We have discussed a number of problems with and challenges associated with electronic election systems,[2,3] and the fallacies associated with improper practical use of cryptography.[4] Numerous technical issues have been reported on by essentially all researchers and evaluators who have conducted independent assessment of security and integrity of electronic election systems, ranging from the well-known problems with the use of DRE systems (direct recording, electronic) and the security vulnerabilities in all examined stand-alone

[1] One Hundred Seventh Congress of the United States of America. *Help America Vote Act of 2002 (HAVA)*. Washington, DC, 2002.

[2] T. Antonyan, S. Davtyan, S. Kentros, A. Kiayias, L. Michel, N. Nicolaou, A. Russell, and A. Shvartsman. "State-Wide Elections, Optical Scan Voting Systems, and the Pursuit of Integrity". In: *IEEE Transactions on Information Forensics and Security* **4** (4), pp. 597–610, 2009.

[3] A. A. Shvartsman, A. Kiayias, L. Michel, and A. Russell. "On the Security and Integrity Issues of Optical Scan Voting Systems". In: *County of Nassau Board of Elections against State of New York, New York State Board of Elections*. Supreme Court of the State of New York, County of Nassau, pp. 1–23, 2010.

[4] S. Davtyan, A. Kiayias, L. Michel, A. Russell, and A.A. Shvartsman. "Integrity of electronic voting systems: Fallacious use of cryptography". In: *Proceedings of the ACM Symposium on Applied Computing* (SAC 2012). ACM Press, 2012, pp. 1486–1493.

systems to severe problems in Internet voting systems.[5] In addition to raising concerns about the dependability of electronic voting systems and suitability of adopted solutions, the ensuing reexamination led a number of states to abandon their chosen solutions and switch technology altogether at a remarkable cost to the U.S. taxpayer,[6] e.g., most recently in Maryland,[7] where the decision was made to migrate to Optical Scan technology, which emerged as the safest option due to its reliance on voter generated paper audit trail.

Whereas researchers and evaluators exposed a number of vulnerabilities in the design, implementation, use of cryptography, logging capabilities, and communication software of voting systems that an attacker could exploit to alter the result of an election, it is not surprising that the bulk of these problems can be traced back to the divergence from, and the ignorance of, established results in algorithmics, verification, validation, cryptography, and sound engineering practices.

Electronic Poll Books. It is interesting that while the premature deployment of immature technology resulted in numerous documented cases of serious problems with voting terminals and election management systems, another component of the electoral process—provision of registered voter lists—still broadly relies on perilously inadequate manual solutions. For example, the use of a questionable manual process caused poll opening delays, long lines, and errors in the Connecticut capital Hartford in the 2014 elections, possibly disenfranchising numerous voters.[8] The emerging computerized solutions, "electronic poll books", are the third component of a voting system: their basic purpose is to ensure "One voter, one vote." Naturally, the integrity and security of this component is paramount to the entire electoral process.

Any electronic poll book solution is an inherently distributed computer system: it must permit concurrent check in of voters while not containing single points of failure, and it must incorporate a consistent replicated storage facility maintaining lists of voters and allowing updates to voter records. Yet, apparently vendors are again rushing to produce "electronic poll books" based on naive premises

[5] S. Wolchok, E. Wustrow, D. Isabel, J. A. Halderman: "Attacking the Washington, D.C. Internet Voting System." In: *Financial Cryptography* 2012: 114-128.

[6] P.J. Peisch. "Procurement and the Polls: How Sharing Responsibility for Acquiring Voting Machines Can Improve and Restore Confidence in American Voting Systems". In: *The Georgetown Law Journal* 97.877 (2009), pp. 877–915.

[7] E.Cox, "New Voting Machines Finally on Horizon: Seven years later, Maryland finally buying voting machines with a paper trail." *The Baltimore Sun*, December 16, 2014.

[8] A. Aponte, J. Cruz, C. Jennings, D. MacDonald, and S. Wooden. Committee of Inquiry Report of Factual Findings. Tech. rep. City of Harford Court of Common Council, 2015.

and software fraught with deficiencies, e.g., as illustrated by a recent patent grant[9] (the analysis of this patent is outside of our current scope). The desire to modernize and ease the administrative process can again lead to the premature adoption of severely deficient solutions that would ultimately fare no better than the doomed Direct Recording Electronic (DRE) voting terminals.

Yet, the distributed computing community contributed elements of solutions and approaches applicable to the underlying problem that revolves around a collection of computing devices reliably and securely maintaining distributed and replicated databases (i.e., "voter lists") in the face of equipment failures. Equally importantly, the research in distributed computing identified a number of problems that are notoriously difficult to solve or that even cannot be solved in the most general setting. The lack of adoption of existing techniques and certain obliviousness of the non-trivial challenges in providing solutions could be attributed to a lack of awareness from practitioners or, equally likely, to the difficulty of specializing and implementing the known approaches to their application domain.

In this paper we focus specifically on the *distributed system aspects* of electronic poll book solutions (and we leave for future work other key aspects, such as security and catastrophic failure recovery). We identify the most important requirements that any solution must satisfy, being a distributed system. We also present several facts from the theory of distributed systems that make implementation of electronic poll book solutions a challenging task.

Document Structure. In Section 2 we describe electronic poll books, their attributes as distributed systems, and the resulting technical questions. In Section 3 we present the requirements that specifically address the distributed nature of electronic poll books. In Section 4 we presented selected facts from the distributed systems research that make it challenging to implement electronic poll books. We conclude in Section 5.

2 Electronic Poll Books as Distributed Systems

In this section we describe the basic setting for poll books, the technological challenges and specific questions regarding electronic poll book implementations.

[9] T. Iredale and K. Clark. System and method for synchronizing electronic poll book voter databases. U.S. Patent US 8812594 B2. 2014.

2.1 The Manual Process

Consider the objectives of officials running an election at the scale of a precinct. Prior to the election, officials are accumulating in a database the collection of registered voters throughout the precinct. On Election Day, a paper listing is printed for each polling station indicating which voters are expected. If electoral procedures permit voters to register "on site", the official must also record individuals who desire to vote but are not on the voter list. As ballots are issued to voters (after they prove their identity), their names get crossed off the list. Additionally, absentee voters who were previously issued a ballot must be accommodated if they decide to vote in person (in some jurisdictions this is permitted). The process is meant to help achieve "one voter one vote". Naturally, a rogue voter can go from polling station to polling station and attempt to vote several times in this way. In such a case, since the authorities collect voter credentials for on-site voting, they would have a legal recourse. From the procedural and safety standpoints, no further significant weaknesses exist with a paper solution.

Yet, a paper process is slow, error prone, and work intensive before, during and after the election as the crossed-off voter information must be collated and re-encoded in the voter database. An electronic solution is appealing to streamline the process, but it also creates many difficulties.

2.2 Distribution and Consistency: Immediate Challenges[10]

Shared storage services are at the core of most information-age systems and electronic poll books (EPB) are not an exception. Imagine an EPB implementation based on a storage system that is implemented as a central server. The server accepts requests from EPB devices to perform operations on its data objects (e.g., voter records) and returns responses. While this is conceptually simple, this approach already presents two major problems. The first is that the central server is a performance bottleneck. The second is that the server is a single point of failure. The quality of service in such an implementation may degrade as the number of users grows, and the service becomes unavailable if the server crashes. Thus the system must, first of all, be *available*. This means that it must provide its services despite failures within the scope of its specification, for example, the system must be able to mask certain server and communication failures. The system must also support multiple concurrent accesses without imposing unreasonable degradation in performance. The only way to guarantee availability is through *redundancy*, that is, to use multiple servers and to replicate the data among

[10] P.M. Musial, N.C. Nicolaou, A.A. Shvartsman: "Implementing distributed shared memory for dynamic networks." *Communications of the ACM* **57**(6): 88-98, 2014.

these servers. Moreover, the replication must be done at geographically distributed and distinct network locations, where the disconnection or failures of certain subsets of data servers can be masked by the system.

It is also critically important to ensure data longevity. A storage system may be able to tolerate failures of some servers, but over a long period it is conceivable that all servers may need to be replaced, because no servers are infallible. Additionally, it may be necessary to provide migration of data from one collection of servers to another as the needs dictate. The storage system must provide seamless runtime migration of data: one cannot stop the world and reconfigure the system in response to failures and changing environment.

A major problem that comes with replication is consistency. How does the system find the latest voter record if the data is replicated? This problem was not present with a central server implementation: the server always contains the latest value. In a replicated implementation, one may attempt to consult all replicas in search of the latest value, but this approach is expensive and not fault-tolerant as it assumes that all replicas are accessible. In any case, none of the implementation issues should be a concern for the users of the system. What the users should expect to see is the illusion of a single-copy object that serializes all accesses so that each operation that reads the object returns the value of the preceding write or update operation, and that this value is at least as recent as that returned by any preceding operation. This notion of consistency is formalized as *atomicity*[11] or, equivalently, as *linearizability*[12]. In the rest of this paper we assume this, most desirable, notion of consistency.

2.3 Specific Technical Questions

Here we present some immediate questions that any EPB solution must address as a distributed system. These questions pose non-trivial challenges in implementing functional and dependable solutions.

1. Multiple electronic devices are needed to allow concurrent check in of voters. How does one ensure that failure of a single device does not interfere with the operation of other devices? How are faulty devices removed from the system? How are new devices introduced to either replace faulty devices or to deal with high voter turnout?

[11] L. Lamport. “On Interprocess Communication. Part I: Basic Formalism.” *Distributed Computing*, **2**(1):77–85, 1986.

[12] M. P. Herlihy and J. M. Wing. “Linearizability: A Correctness Condition for Concurrent Objects.” *ACM Transactions on Programming Languages and Systems,* **12**(3):463–492, July 1990.

2. If the hardware of electronic devices can fail, this raises questions about the status of the information (about the election and voters) accumulated in each device. How is the information recovered? Passed on to other devices? Replicated in real-time? How does one transfer the content to a "spare device" to quickly resume?
3. Where is the relevant information (voter lists) held? On each device? On a server nearby on a local area network? In the cloud for more reliability? Depending on the answer, one must question what to do in case of network failure. What to do if the network connectivity is lost? Experiences delays? How does one deal with transient (or prolonged or even permanent) network partitions?
4. Any real-time replication implies the execution of some protocol that may itself be subjected to perturbations, e.g., denial of service attacks. How to guarantee the validity and legitimacy of messages exchanges among the participating devices?
5. When multiple devices are used to enter new voters or cross off voters upon handing out ballots, how does one maintain a coherent view of the world guaranteeing consistency and enforcing the rule of voting at most once?

Given the none too complimentary state of the electronic election systems, there is the natural concern that existing and emerging offerings for electronic poll books might side-step the majority of these questions leaving potential customers with brittle systems that suffer from major shortcomings and that perform adequately only in friendly environments.

3 Requirements for Electronic Poll Book Solutions as Distributed Systems

The State of Connecticut recently published requirements for electronic poll book systems.[13] We extract from this specification and present several requirements that specifically address the necessarily distributed nature of electronic poll book solutions. We begin by stating several definitions for the terms used in the requirements.

[13] *Connecticut Electronic Poll Book System Requirement Specification V1.0*. Approved, Denise W. Merrill, Connecticut Secretary of the State, March 6, 2015. http://www.sots.ct.gov/sots/lib/sots/ElectionServices/Handbooks/e-poll-book-system-requirements-1_0c_(2).pdf

Definitions of Terms

- ***Electronic poll book system (EPBS)*** – A collection of hardware and software including at least one configured *electronic poll book* and aiming to implement electronic poll book functionality that satisfies the requirements stated in this document.
- ***Electronic poll book (EPB)*** – A component of the *electronic poll book system* that includes a user interface device and that is to be used by a poll worker to view and update voter registration records.
- ***Electronic poll book system configuration (EPBSC)*** – A physical instance of an *electronic poll book system* with all its components configured for use. An *electronic poll book system configuration* consists of peripherals (e.g., printers, scanners, etc.) and a set of configured, networked *electronic poll books*. An *EPBSC* may also contain auxiliary servers.
- ***Voter record*** – The *voter registration record* and *voter activity record* of a voter.
- ***Local voter database*** – A collection of all *voter records* specific to a jurisdiction (e.g., a precinct or district). The initial state of the *local voter database* is compiled and certified by the relevant authority. Poll workers make updates to the *local voter database* throughout the election by using the *electronic poll book system* to reflect ongoing voter activity within the jurisdiction.
- ***Voter list*** - A printable, exportable, and human-readable representation of the *local voter database*.
- ***Completed update*** – An update to a *voter registration record* is *completed* if a query for said *voter registration record* on any active *electronic poll book* within the *electronic poll book system* returns the same data.
- ***Quiescent*** - The *electronic poll book system* is *quiescent* if all user-initiated updates have completed at all *electronic poll books*.
- ***Reconfigure EPBSC*** – Configuring, adding, or removing any of the *electronic poll book configuration's* peripherals, *electronic poll books*, or auxiliary servers.

Requirements Relevant to the Distributed Nature of the System

Here we present and abridged set of requirements that specifically deal with the distributed nature of any comprehensive electronic poll book solution. Broadly speaking, these requirements are formulated to ensure the following.

- Fault-tolerance: The system must not contain any single points of failure, and should a failure be encountered (up to a design limit) it must not prevent the rest of the system from operating. Main types of failures are the failures of the physical system components or the software in these components, and communication failures.
- Service availability: The system must be able to provide the required service in the face of adversity and perturbations (again, up to its design limits).
- Data consistency: The data contained within the system (e.g., voter records) must be viewed consistently following any changes to the data.
- Data survivability: No data may be lost if certain components of the system fail (up to its design limit).
- System reconfigurability: The system must enable faulty components to be removed and/or replaced without requiring halting or restarting the overall system.

We now state the most relevant requirements[14] in an abridged form.

- **AR-2:** No single point of failure.

  Description: The *electronic poll book system* must be designed to tolerate any single point of failure scenarios.

- **AR-1:** At least three *EPBs* in an *EPBS*.

  Description: An *electronic poll book system* must support at least three (3) *electronic poll books* in a single polling location. Each of the *electronic poll books* must be usable concurrently. Should one of the *electronic poll books* become inoperable, the operation of the remaining *electronic poll book* or *electronic poll books* must not be affected.

- **FR-1:** Adding a new *EPB* to the *EPBS*.

  Description: The *electronic poll book system* must provide means for the integration of an additional *electronic poll book* into its configuration at any point throughout the election without requiring a shutdown or a restart of the *electronic poll book system*.

- **FR-2:** Removing an *EPB* from the *EPBS*.

[14] *Ibid.*

Description: The *electronic poll book system* must provide means for the exclusion of an existing *electronic poll book* from its configuration at any point throughout the election without requiring a shutdown, or restart of the *electronic poll book system*. This action does not require physical access to the *electronic poll book* that is to be excluded.

- **FR-21:** One voter / one vote within *EPBS*.

  Description: The *electronic poll book system* must guarantee that within an *electronic poll book system configuration* a voter can be checked in at most once during normal connectivity.

- **RR-1.1:** Voter check-in during interruption of connectivity.

  Description: In the event of a temporary interruption of connectivity within an *electronic poll book system*, the *electronic poll book system* must permit a voter to check-in.

- **RR-1.2:** Upon restoration of connectivity.

  Description: In the event of a temporary interruption of connectivity within an *electronic poll book system*, the *electronic poll book system* must automatically restore *voter list* consistency across the *electronic poll books* after connectivity is restored.

- **RR-1.3:** Identify double voting.

  Description: In the event of a temporary interruption of connectivity within an *electronic poll book system*, the *electronic poll book system* must identify voters that have been checked in at two or more different *electronic poll books* during the interruption of connectivity.

- **RR-5:** *Local voter database* replicas.
  Description: Within the *electronic poll book system* there must exist at least two replicas (logical or physical) of the *local voter database.* These replicas must be stored in distinct physical storage components. (Note: while this introduces replication, together with AR-1 and RR-1.1 the number of replicas may need to be higher.)

- **RR-6:** *Local voter database* replica consistency.

  Description: If the *electronic poll book system* is in a *quiescent* state all replicas of the *local voter database* must be logically consistent.

- **RR-7:** Operational consistency.

Description: Any update to a *voter record* or to any other data pertaining to the election completed on one *electronic poll book* must be seen as *complete* on all other *electronic poll books*.

We believe that these requirements are quite intuitive, and we consider them necessary for any implementation of an electronic poll book system. Next we identify several results from the field of distributed computing that make it challenging to satisfy these requirements.

4 Electronic Poll Books in the Light of the Distributed Systems Theory

In this section we describe several results from the distributed systems theory that stress the needs for careful and diligent design in developing electronic poll book solutions. At first glance, the results we cite cast a pessimistic view on one's ability to build dependable systems that need to coordinate their activities in non-trivial ways or that maintain replicated shared data for which consistency is paramount (e.g., if a data object is changed, then the following read of the object value must reflect the change). However, this does not mean that one cannot build reliable and usable electronic poll book systems. In order to succeed, one needs to understand the theoretical limitations and to make sensible assumptions about the nature of failures, communication, and asynchrony. The main point here is that any system that claims to provide a solution that is able to deal with requisite adversity and perturbations in the computation medium, but that is oblivious of these known results, is to be suspect.

In what follows we do not explicitly cross-reference the requirements from Section 3, but, as we have indicated earlier, the small selected set of requirements deals collectively with the issues of fault-tolerance and availability (AR-1, AR-2), communication (RR-1.1, RR-1.2), agreement and consistency (FR-21, RR-1.3, RR-6, RR-7), and survivability (FR-1, FR-2, RR-5) of the shared data on which the electronic poll book function relies. (We refer the reader to the full requirements document[15] for additional details.)

Lastly, here we focus on the negative (impossibility) research results. Although specialized practical solutions exist for certain modified versions of the problems given here, we do not present them: not only the solution space is very large, but more importantly, it is the duty of responsible system designers to investigate relevant solutions when they will have started gaining the necessary insight.

[15] *Ibid*.

4.1 Consistent Data Store with Device Crashes

Any electronic poll book system must be able to tolerate benign failures of individual devices, specifically, a failure where the faulty device stops at an arbitrary instant of time and does not perform any further actions. Such benign failures are known as *crashes*. For example, a polling place may have several devices used to check in voters. A crash of such a device must not prevent other working devices from functioning, and the crash must not destroy the consistency of the shared data maintained by the system (of course the data must be replicated for survivability). For example, if a voter was successfully checked in, then all operating devices must agree that this is the case, regardless of a crash. If this cannot be guaranteed, then an ill-motivated voter may attempt to vote more than once. Suppose the type of data we are interested in is consistent read/write data.[16] This is a basic data type, much simpler than data types that support more complicated read-modify-write operations. It turns out that any system of devices implementing such objects can tolerate the crashes of only a minority of the devices.[17,18]

> *A system of N processors cannot implement a consistent read/write object where all object access operations terminate (complete) in the presence of F crashes if* $N \leq 2F$.

This means that to tolerate a crash of just one device, the data must be replicated at three devices. To tolerate two device crashes requires replication at five devices, etc. In particular, any poll book solution that replicates its data in two locations and that claims to tolerate a single crash during the operation of the system cannot possibly be correct.

4.2 Coordinated Action with Link Failures

It is clear that multiple devices are necessary in any implementation of an electronic poll book system. If there are multiple devices providing some service, they must do so consistently. Suppose several devices (say, individual poll book devices) need to agree on a common course of action, e.g., by deciding on a value that indicates what action to take. This is known as the *agreement problem*, and the correctness conditions for a solution are as follows:

---

[16] Recall that an implementation of a data object is consistent (atomic or linearizable) if the users that access the object (perform reads and writes) are presented with an illusion that there is a single copy of the object that is accessed sequentially by the users regardless of how the object is implemented in the underlying distributed system.

[17] G. Bracha, S. Toueg: "Asynchronous Consensus and Broadcast Protocols." *Journ. ACM* **32**(4): 824-840, 1985.

[18] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, R. Reischuk: "Renaming in an Asynchronous Environment." *Journ. ACM* **37**(3): 524-548, 1990.

(*a*) *Agreement*: no two devices agree on different values,

(*b*) *Validity*: if all devices propose the same value, then this is the only possible agreement value,

(*c*) *Termination*: all non-faulty devices eventually decide.

Now suppose that there are just two devices that never fail, but communication can be unreliable, e.g., messages can be lost because of failures or interference. If this is the case, one of the oldest results in distributed computing tells us that there is no protocol that always solves this agreement problem.[19]

> *There is no algorithm that solves the coordinated action problem for two processors that communicate using unreliable messaging.*

Needless to say, if the problem cannot be solved for two devices, it cannot be solved for any larger number of devices. Of course, this problem still needs to be solved in real system. This is normally done by strengthening the assumptions about the model of computation or by relaxing the problem requirements.[20] For example, this can be done by limiting the types of failure that the system tolerates and by stating guarantees probabilistically, thus allowing a system to be incorrect with very small probability. (Similar approaches can be applied in solving other problems we describe in the sequel.) Incidentally, this problem is known as the "Two Generals Problem" in the literature. Here, two generals must launch a coordinated attack, lest they be defeated one at a time by the opposing force. The generals communicate by messengers that can be intercepted or destroyed.

4.3 Availability and Consistency in the Presence of Network Partitions

All devices providing the electronic poll book function to its users must present a consistent view of the underlying data shared by the system. Recall that the service must not be centralized to eliminate any single points of failure. Thus to ensure system availability it must be distributed. If the system implementation is distributed, it must rely on some network for communication among its components. The implementation cannot assume that communication is always reliable; in particular, network failures may isolate some of the devices in the system, i.e., the network may partition. Clearly it is desirable for the service to be available (any request to access shared data receives a response) and

[19] J.N. Gray: "Notes on Data Base Operating Systems." *Advanced Course: Operating Systems*, Lecture Notes in Computer Science 60, Springer, pp. 393-481, 1978.

[20] N.A. Lynch, *Distributed Algorithms*, Morgan Kaufmann, 1996 (Chapter 5).

consistent (atomic/linearizable). However, the well-known "Brewer's conjecture" posits that it is not possible to simultaneously guarantee consistency, availability, and partition-tolerance.[21]

> *It is impossible for any distributed service implementation to provide the guarantees of (i) consistency, (ii) availability, and (iii) partition-tolerance.*

This statement can be made more specialized for read/write objects as follows.[22]

> *It is impossible for any distributed service implementation of shared read/write data objects to guarantee (i) consistency, and (ii) availability, if the underlying asynchronous messaging system allows for message loss.*

This means, in particular, that if messages can be lost (e.g., due to jamming or denial-of-service attack on the poll book system), then either the data (e.g., voter records) may appear inconsistent or the service may be unavailable. Interestingly, the above result holds even if after some time the system becomes synchronous, with known delays on the messages.[23]

4.4 Reaching Agreement in the Presence of Crashes and Asynchrony

A polling place with several devices used to check in voters must be able to tolerate at least one crash. The crash must not prevent the overall system from taking coordinated actions. As before, the electronic poll book devices may not be in perfect synchrony with each other, e.g., processing delays and arbitrary timing of actions by the poll officials is likely to introduce some measure of asynchrony. Unfortunately, reaching agreement in the presence of even a single crash may be impossible in all cases for an asynchronous system, even if no message is ever lost. A seminal and venerable result from the distributed computing theory states the following.[24]

> *For an asynchronous system of processors that communicate using reliable channels there is no algorithm that solves the agreement problem and that guarantees termination in the presence of a single crash.*

---

[21] Eric A. Brewer. "Towards Robust Distributed Systems." (Invited Talk). *ACM Symposium on Principles of Distributed Computing*, Portland, Oregon, July 2000.

[22] S. Gilbert, N.A. Lynch: "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services." *SIGACT News 33*(2): 51-59, 2002.

[23] *Ibid*.

[24] M.J. Fischer, N.A. Lynch, M.S. Paterson, "Impossibility of distributed consensus with one faulty process." *Journal of the ACM* **32** (2): 374–382, April 1985.

This means that in the most general setting, if a system relies on solving the agreement problem as part of its implementation, there may be some operations that never complete (or that are very slow). Thus, any system that claims that all of its devices are always in some type of agreement on certain values and that has good performance for all operations must make several non-trivial assumptions about the nature of failures and the constraints on asynchrony. If these assumptions are not explicitly stated, then the claims are to be taken with a grain of salt.

4.5 Reaching Agreement in the Presence of Malicious Failures

Given the plethora of malware and viruses that may affect a computer system, an electronic poll book system may also need to tolerate malicious failures of individual devices. Such malicious failures are called *byzantine* failures.[25] Here if a device fails, it does not stop as in a crash, but instead starts behaving arbitrarily, and in particular, it may perform malicious actions. This will be the case if a device is maliciously tampered with, or if it is infected with malware.

For this setting, another seminal result states that a system of three devices (processors) cannot tolerate even a single byzantine fault.[26]

> *A system of three processors cannot solve the agreement problem in the presence of a single byzantine failure.*

Note that this result holds even if the processors are in a complete synchrony with each other and if there are no other perturbations, such as message delay or loss.

For electronic poll book systems this means that a system with less than four devices cannot tolerate even a single malicious failure. The more general result dictates that any system of processors cannot tolerate malicious failures of even a third of the processors.[27]

> *A system of N processors cannot solve the agreement problem in the presence of F byzantine failures if* $N \leq 3F$.

Thus, in any system where the correct devices must reach agreement, the correct devices must outnumber the faulty devices by more than a factor of three-to-one. If this is not logistically feasible

[25] L. Lamport, R.E. Shostak, M.C. Pease: The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst*. 4(3): 382-401, 1982, and M.C. Pease, R.E. Shostak, L. Lamport: Reaching Agreement in the Presence of Faults. *J. ACM* 27(2): 228-234, 1980.

[26] *Ibid*.

[27] *Ibid.*

for a real installation, then the system cannot possibly claim to tolerate tampering with (or theft of) even one device.

4.6 The Problem of Reconfiguration in Dynamic Systems

Thus far we only considered static systems, that is, a system where the universe of devices is fixed in the initial state of the system. Providing electronic poll book solutions only for static systems is clearly inadequate. Consider a polling place with three initial devices. On the Election Day everything proceeds smoothly for a while. However the voter turnout is much higher than expected and the lines are getting long. Now suppose one of the three devices crashes. Even if the remaining two devices are operational and are able to provide the needed services, the lines of voters are getting really long now. A well-designed system must always allow for additional check-in devices to be introduced in order to cope with the faulty devices and the higher-than-expected voter turnout. Needless to say, this must be accomplished without halting the check-in process and without restarting any devices in the system.

The general problem of removing some devices from a system and introducing new devices into the system is known as the *reconfiguration* problem. In our context, we are not concerned with simply adding and removing devices: we need to also make sure that no data is lost and that the new devices are brought up to date with respect to the state of the data. Here all devices must have a consistent view of the state of the system. For a distributed system that is charged with maintaining consistent shared data the reconfiguration operation is described as follows.[28]

> *Reconfiguration is the process of replacing one set of devices participating in a distributed system with an updated set of devices. In this process, the data is propagated from the old set of devices to the new set, and allowing devices that are no longer in the new configuration to safely leave the system. This changeover has no effect on ongoing data-access operations, which may continue to store and retrieve the shared data.*

Development of algorithms implementing reconfiguration is still an active area of research. While there exist algorithms and system implementations that incorporate reconfiguration, e.g., the RAMBO (Reconfigurable Atomic Memory for Basic Objects) framework,[29] it is unknown whether the most general such implementations must necessarily implement reconfiguration with the help of solutions to the agreement problem. Recall that we have already identified several challenges associated with this

[28] S. Gilbert, N.A. Lynch, A.A. Shvartsman: "Rambo: a robust, reconfigurable atomic memory service for dynamic networks." *Distributed Computing* **23**(4): 225-272, 2010.

[29] *Ibid*.

problem. In any case, the solutions to the reconfiguration problem are going to be difficult and fraught with impossibility results akin to those we discussed here.

Regardless of how the reconfiguration of the set of devices is done, any practical implementation of electronic poll books must address the challenge of deciding when to reconfigure. One approach is to leave this decision to the environment, e.g., the users of the system. Access to the reconfiguration service should be available to system administrator to enable reconfiguration based on various policies, such as introducing new device in case of high voter turnout or removing misbehaving devices. For larger installations, reconfiguration could be enacted automatically when a failure of a certain number of devices is detected. This is a more complicated solution, but it has the potential of providing superior quality of service.

Given that this is an active area of research and that solutions to the reconfiguration problem are not routine, one must exercise caution. Any electronic poll book system that claims to provide reconfiguration features without supporting documentation and without rigorous arguments about the system's correctness must be carefully examined before using such features on the Election Day. An available survey[30] contains a broad discussion of reconfigurable storage systems.

5 Discussion

We presented a view of electronic poll book solutions as distributed systems. We cited several requirements that need to be satisfied by electronic poll book systems in order to guarantee fault-tolerance, availability, and consistency. The requirements are necessary for any robust poll book system. These requirements deal only with distributed system aspects of electronic poll book solutions (and they are not sufficient as they do not deal, for example, with security and catastrophic failure issues, which are outside of the scope of the current paper). We then cite several results from the distributed system research showing that it is challenging to build adequate poll book solutions, and that without being grounded in relevant research any solution is likely to be incorrect and provide only an illusion of fault-tolerance. While the research results show that certain key problems in the distributed systems landscape in general have no solutions, one must not be necessarily discouraged. It is possible to build robust systems with the help of insights into the suitable modeling assumptions dealing with computation and adversity, and by moderating the guarantees provided by such systems.

[30] P.M. Musial, N.C. Nicolaou, A.A. Shvartsman: "Implementing distributed shared memory for dynamic networks." *Communications of the ACM* **57**(6): 88-98, 2014.