C-DOG: Census Data Organization for the Government

6.1800 Design Project Report

By Aryan Kumar, Daniel Xu, Roderick Huang

WRAP Instructor: Michael Trice
Recitation Instructor: Karen Sollins

May 8th, 2023

1. Introduction

Census systems are crucial in providing a representation of the population so that governments and policymakers can implement important public decisions. Fictlandia's current census system requires three census forms per year for each level of government–national, state, and municipality. Due to the repetitiveness of these forms, census response rates are low. To combat this problem, we propose **C**ensus **D**ata **O**rganization for the **G**overnment (C-DOG), which (1) allows households to submit just one form per year, (2) pushes appropriate data to authorized users, (3) aggregates collected data into a secure format, and (4) stores all collected data long-term.

The design of C-DOG prioritizes *availability* and *security*. Our primary goal is availability, which means that the census system is accessible to all those who need to use it, even during periods of high load on the system. In addition to the mentioned three levels of government, other organizations also need access to the data including the School Board, Election Board, and researchers. Additionally, we prioritize *security* as it is essential to prevent sensitive data from being tampered with. If data leaks occur due to a lack of security measures, people would immediately lose trust in the system and refuse to take part in the census, defeating the main objective of improving the response rate.

C-DOG achieves these design priorities by assigning machines in the municipalities which are closest to people to manage data submission and transmission. The machines follow a strict tiered priority system (TPS) to perform certain data tasks depending on the system load at that time. This helps to achieve a balance between sending data to organizations that need it in a timely manner and storing data for long-term usage. Through the use of public encryption keys, we ensure that the data is sent securely and stored securely in the distributed database.

In Section 2, we discuss the main functional components of C-DOG and how they are connected at a high level. Afterward, in Section 3, we delve deeper into each component to demonstrate the specific design choices made to follow our design priorities. In Section 4, we evaluate our design alongside applicable use cases of the census system.

2. System Overview

C-DOG relies on the following main functional components: data collection, distribution, storage, and access. Within each component, measures are taken to ensure *availability* and *security* of the data. In Figure 1, we show an overview of the modules of our design and how they're all connected.

- *Data Collection*: A user inputs a household census form into a physical machine within the user's municipality. Depending on the load on the system and the type of form submitted (paper or online), the municipal machines follow the tiered priority system (TPS) that will prioritize certain tasks over others. For instance, we prioritize supporting online form submission over uploading data to account for low user retention rates in the system. It has been deemed more vital to ensure the submission server is available to a user whenever necessary so as to not lose valuable data that could be collected.

- *Data Distribution & Transmission*: The municipal machines continue to follow the TPS to prioritize what data should be transmitted. Distribution occurs when the load to take in submission forms is relatively low. Prioritizing security, encryption keys are utilized to secure the data when it is being transmitted. C-DOG also employs TCP to achieve reliable and in-order transport. However, in the event of a network failure, each municipal machine has a queue that can re-transmit failed sent data to further boost fault tolerance.
- *Data Storage*: The storage of C-DOG at the municipal level is divided into two areas: local storage and virtual storage. The local storage consists of a distributed local file system that will store the paper forms along with aggregate data and a local distributed database that will store the data over all years. The virtual storage consists of a distributed database that is accessed through virtual machines that connect to each physical machine and stores a copy of all the data to improve availability. Tables to organize the database and folder hierarchies to organize the file system are discussed in more detail in 3.2. The national database, containing national-level data pushed from the municipal machines, is constructed similarly with tables and also discussed in 3.2.
- *Data Access*: Depending on the user and where the data is stored, authorizations are put in place to ensure that the user only gains access to appropriate data through access control lists (ACL). For requests to individual census records, in the event of read failures to the local storage to access the data, the virtual storage acts as a fallback to ensure the availability of the data.

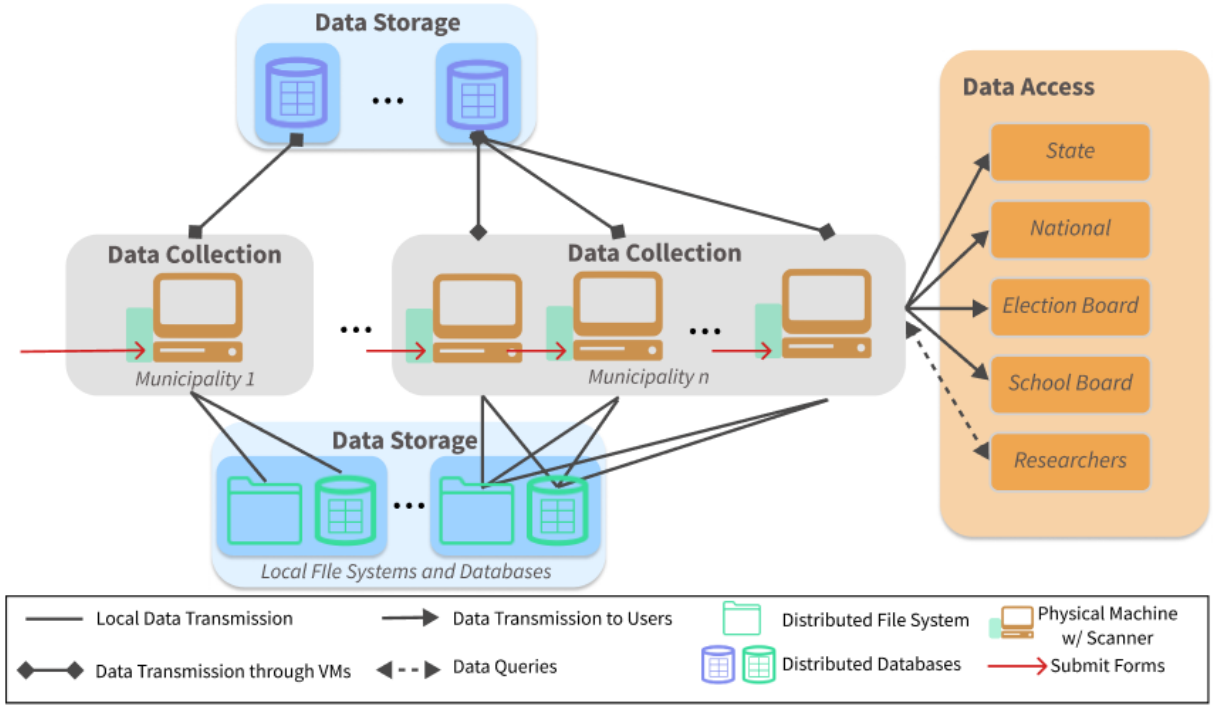

Fig. 1 Illustrates the main functional components of C-DOG

3. System Design

3.1. Data Collection & Processing

The municipal machines will be responsible for collecting and processing all local census records. Depending on the type of form received and the users that need the data, varying courses of action are taken to address each case.

3.1.1 Collecting Online & Paper Form Data

Census data collection begins at the municipal level, where municipalities will collect data in one of two ways: paper forms or online forms. The online forms will be hosted on municipal machines, with each municipal machine capable of supporting a maximum of 125 users completing the form at a time. In the event that there are more than 125 concurrent users, those users will be given the option to wait in a queue.

To maximize census participation, we allow users to submit partially complete forms to make form submission as easy as possible for them. More specifically, the user can submit a partially complete form with certain identifying information complete (i.e. full name and address). The municipal machine that receives this form will immediately write this partially complete record into the shared local database, in a table consisting of all of the partially complete municipal census records for this given year. This way, any machine (including a different machine in the same municipality) can pull the data later when the same user logs on, and allow the user to resume from where they left off. At the end of the census collection period, since this table will no longer be in use, we will discard it for storage purposes.

Paper forms, on the other hand, can be filled by people at their own leisure, and mailed to the municipality whenever. Once received, the forms will be scanned by an optical reader attached to a municipal machine, which will generate both a pdf scan of the paper form and a record containing the data in the form. This record will be inserted into the shared municipal database, and the paper form scan will be stored on a municipal machine.

3.1.2 Processing & Distribution of the Data

Upon receiving an online form, the municipal machine will immediately insert the record into the local database and write it into its transmission queue (described more in 3.4) which exists on the machine's persistent storage, to be sent to the various users of the data (school & election boards, states, etc), paired VM, and the national cloud service. This ensures a failure in the municipal machine does not wipe out the census record, and we can resend it. We also ensure we only transmit the relevant subset of the municipal census record to the entity, to comply with legal requirements and ensure the privacy of our citizens.

When the paired VM receives the census record, it will insert the record into its database. The national cloud service machine that receives the census record will similarly insert the record into the national

database. We discuss data storage further in 3.2. The census data is also sent to various users, but what they do with the data is beyond the scope of our system. The networking protocol (which we discuss further in 3.4) requires that each of the receiving machines transmit an ACK back after successfully writing the entire received file, allowing the municipal machine to re-transmit the data in the face of various failures and ensuring that the users and national government receive the data as intended.

3.1.3 Tiered Priority System (TPS)

Since the municipal machines are central to the census system's data collection abilities but have limited computational resources, we will use a tiered priority system (TPS) to determine which tasks the municipal machines should attempt to perform:

1. Support online form submission
2. Queries for municipal census data
3. Uploading paper forms onto the local machines
4. Uploading census data to national government's data centers and to users

We make this choice based on our system's emphasis on availability. We prioritize being able to accept new census data and handle requests for local census data efficiently above having a completely up-to-date census system. If the former services are unavailable, we will lose several users' worth of census data and the municipalities will not be able access data they need to provide services to the populous directly.

Based on this tiered system, we will handle computation as follows. During periods of greater load on municipal machines (from supporting online form submission or from being queried for data by the municipality), they will withhold sending any accumulated census forms to the users and national data centers, and withhold scanning any paper forms until the load dissipates. When the municipal machines are eventually under-utilized, particularly during night hours when no census data will be inputted, they will scan any accumulated paper forms and upload any outstanding census data to the national data centers and the users.

3.2. Data Storage

3.2.1 Record Data Storage

A municipality's census data will be stored on their municipal machines' databases and file system. Specifically, the database will store a table for each year's municipal census data, where the rows are the individual census records and the columns are the fields of the record. In addition, the municipal machines will store each year's municipal census data in a file, named `census_data_{year}.txt`. During periods when census data is actively being collected, this file will be recompiled from the local database every day. The municipal machines will each read a roughly evenly sized chunk of the local database table for the current period, list the read records in a file, and then merge them together. In the rare event that an individual municipal machine fails while performing its task, we will discard the new file that is being compiled, and we will keep the file compiled the previous day.

This file compilation process will be done during the night, as we expect the municipal machines should have little to no load during this time. It should take roughly 100,000 records/machine * 0.07 seconds/read record = 1.94 hours to perform, and this file will help us quickly compute aggregate summary statistics at the municipal level as will be discussed in 3.3. We make a deliberate choice to update this file only once a day, since it is a meet-in-the-middle approach. On one hand, we want summary statistics to be relatively up-to-date during the active census period, but on the other hand, we do not want to overburden our municipal machines with significant extra computation from continually recompiling this file. The paired VMs for a municipality will store that municipality's data in their database and file system similarly.

We emphasize that in the above scheme for municipal data storage, each municipality is storing only its own census data. Having this modular approach to storage ensures different municipalities cannot access each other's data, which protects the privacy of their respective citizens, and has security benefits since each citizen's data is only stored in one place at the municipal level.

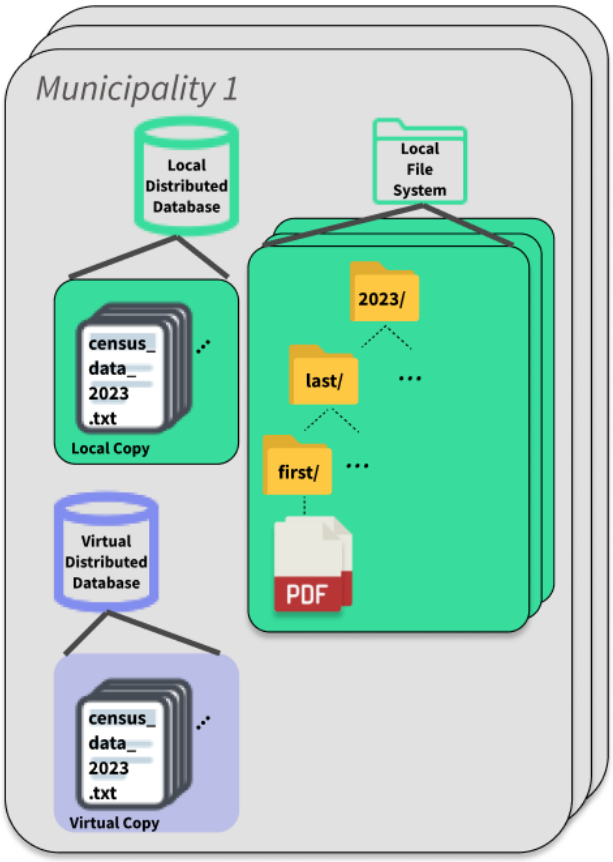

Fig 2. Illustration of the Municipal Data Storage as to how paper and online forms are stored in the local distributed database, virtual distributed database, and local file system.

The national government will store census data on the distributed database that runs on top of the cloud service's machines. Specifically, the national database will have a table for each year's national census data, where the rows are individual records and the columns are the census fields. Note that the national government receives and stores only the subset of the municipal census data that pertains to them. In addition, the national cloud service's machine will each store a file with summary statistics for each year's national census data. During the active census period, this file will be recomputed each night similarly to how it was done at the municipal level. The 3000 machines in the national cloud will each read through a similarly sized chunk of the database table for the current year, compute the relevant statistics in a file, and then merge the results when all of the machines are done. This should take approximately (300,000,000 records) * (0.07 seconds/record read) / 3000 machines = 1.94 hours to do. This file will help us quickly serve requests for summary statistics.

3.2.2 Paper Form Storage

A municipality's paper form scans will be stored on their municipal machines' shared file system. Each paper form scan will be stored in a folder of the following format:

`{census_year}/{last_name}/{first_name}`

This folder will contain the pdf scan files of everyone with the given first and last name who submitted a paper form for the census that year.

We emphasize that we only utilize the municipal machines to store the paper forms, and we do not utilize the national government's cloud service at all for this. We understand that this further constrains the amount of space our system has to store paper forms, which can thus decrease the longevity of our system, but the pdf scans contain some data pertaining only to the municipal census, which the national government should not have access to. Moving even some of the municipal census data to the national cloud service could very well be illegal, and diminish citizens' trust in our system.

3.3. Data Access

The municipal government will primarily query the municipal machines for municipal data. If the municipal machines are overburdened or are inaccessible for any reason, requests can also be sent to their paired VMs as they should have an identical copy of the census data (though possibly slightly not up-to-date during the active census period, due to lags in sending the data).

For individual census record requests, the municipal machines will read the record(s) from the appropriate table of the municipal database, and then serve it to the requestor. For requests for summary statistics for a particular year, the municipal machine will read the file with the census data for that particular year and compute the statistic and serve it. We read the data from the file rather than the database since file read rates are 7000 MBps, and the municipal census data for a given year should take about ~80mb to store per 100,000 people in the municipality. Assuming a municipality has ~500,000 residents, the summary statistic should take only 5 * 80mb / 7000 = 57ms to calculate and serve to the requestor. Queries for municipal data received by their paired VMs are handled similarly.

The national government can access data from the national cloud service similarly. For requests for individual census records, the cloud machine will read the appropriate record(s) from the national database, and serve the request. For requests for summary statistics, the cloud machine will read the file containing the pre-computed statistics, and serve the request.

As previously mentioned, the various users of our system including school & election boards, states, etc will have the relevant data sent to them using the network protocol we discuss in section 3.4. The data is sent to these users as early as it can be transmitted, which has several advantages. For instance, the users get data in close to real time as the data is being collected, which can allow them to begin processing the data and start on their tasks as soon as possible. In addition, this approach decreases the probability the users will face network cloggages from census data being sent to them all at once. It also decreases the probability our system will not be able to send the data to the users in time, as we begin transmitting data immediately rather than after the March 1st collection deadline.

The researchers interacting with our census system can query for national or municipal level summary statistics on the census data, by querying the respective national or municipal database. The queried system will perform the authentication check as described in part 3.6. For state level summary statistics, the researchers will need to interact with the state responsible for that data, since this is outside our system's domain. Since states manage their census data on their own, they are responsible for deciding if the researcher should have access to the state's summary census statistics and if so, decide on how it should be served to them.

3.4. Data Transmission Protocol

Machines within our census system will be sending files and data amongst themselves quite frequently. Thus, we need to design a protocol to ensure the complete and reliable delivery of data, especially in the face of machine failures and network outages.

We implement our networking protocol between machines as follows. At the application layer of the sending machine, the files to be transmitted will be placed into a queue that will exist on that machine's persistent storage (i.e. its file system). Whenever the machine is underutilized according to our tiered priority system, the protocol will attempt to send over the files. The application layer protocol will break each file into a series of packets to be sent over TCP, and will issue an EOF (end of file) to communicate to the receiving machine the entire file is sent. On the application layer of the receiving machine, the packets will be accumulated into the file they constitute. Only once the entire file is received will the receiving machine write it to persistent storage (i.e. a database or file system), and issue an application layer ACK back to the sender once the write is completed. If the sender receives an ACK back before the timeout period, it will take the file off its queue. If it does not, the file will be placed at the back of the queue to be sent later.

This protocol ensures fault-tolerant file transfer. Even if the network or a sending/receiving machine fails in between a file's transmission, the protocol ensures we can retry sending the data. In addition, it also ensures that such a failure would not leave the receiving machine in an inconsistent state, with a partially received file written to disk, since we do not write a file until it is completely received.

The protocol does add additional overhead, in terms of the storage space taken by the queue, and the ACKs sent over the network. However, we expect the queue to take marginal storage on any machine. For example, the municipal machines (which transmit the most data in our system) will each expect to receive roughly 80 MB of online form data and roughly ~15.3 GB of paper form scans per year. Even if the network between the municipal machine and national cloud fails for a few days, we should expect to store a maximum of a few gigabytes of data in the queue, and this protocol moreover ensures the municipal machine can still collect census data and eventually send it. Furthermore, due to the very high network speeds in our system, we do not believe network bandwidth will be a bottleneck in our system's performance, and we prioritize our system's data being accurate and complete through the ACKs. Incorrect/incomplete data that is served marginally faster is far less valuable than accurate data that takes slightly longer to be sent due to a slightly slower network.

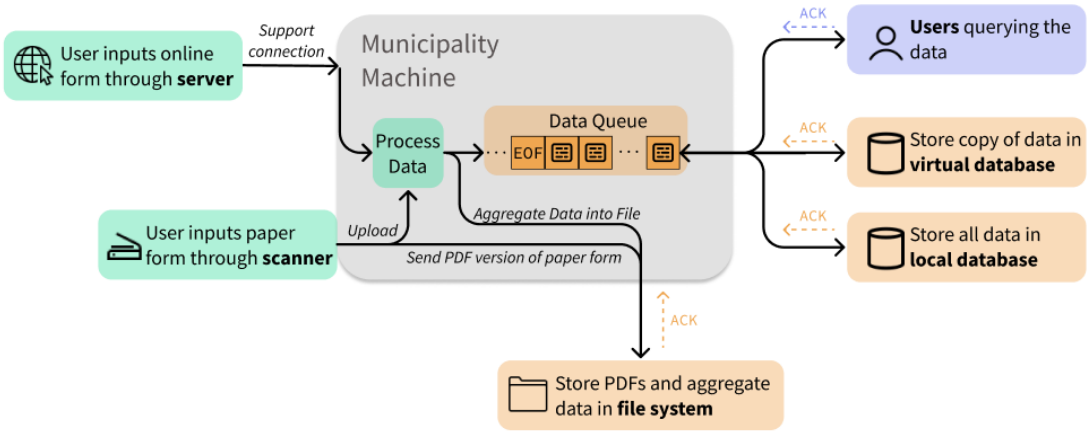

Fig 3. Illustrates the data transmission protocol. It displays forms of user input, data distribution, and

3.5. Data Security and Encryption

We choose to encrypt and sign all data and messages sent over the network to prioritize security. To perform the encryption, every server and user involved in the system has a distinct private key and public key pair. The list of all public keys is distributed through a secure channel at the creation of the system. Data is encrypted using the receiver's public key and the receiver decrypts the data using the private key.

In order to ensure the validity of the message, the sender signs the messaging with a Message Authentication Code (MAC) generated by the private key. Anyone with the sender's public key can verify that the signature must have been created by the sender. Using asymmetric cryptography, all messages can be encrypted and validated. Even though encryption will make the file size 3x larger and take 3x longer to send over the network, we made this design choice because the privacy of our users is a much higher priority than performance. A failure to ensure the privacy of user's data would lead to a lack of trust in the system and a lower census participation in the future.

In addition, requests for data are verified against an access control list (ACL) that is stored on each national and municipal server. The ACL is a file that stores a list of privileges associated with the public key of each user. This will ensure that a malicious actor cannot spoof requests for data they don't have access to. An important security goal is to protect user data from being exposed to unprivileged users. C-DOG is designed so that each jurisdiction stores its own data which prevents the vulnerability that municipalities or states could access records outside their city boundaries. Additionally, the national and state government cannot access details of the census record that belong only to the municipal records. In addition, researchers can access pre-computed statistics such as average age, gender ratio etc. but not individual census records, protecting individual citizen's privacy.

4. Evaluation

With the key components of C-DOG explained in detail, we evaluate it from a quantitative and qualitative perspective.

4.1. Functional Components

In this section, we evaluate each functional component. In particular, we analyze the communication overhead of our system and what our system gains from our custom network protocol in 3.4. We discuss how long operations take to read data and what areas of our system are limited.

4.1.1 Data Submission

When there is a high load on the census submission system, the amount of wait time can decrease the likelihood of the user completing the census form, leading to valuable data loss. As explained in 3.1.2, TPS solves this problem to some extent by prioritizing the support of online submission. Assuming an average daily load of users for a 2-month period,

$(80\% \cdot 100,000\ people\ /\ machine) \div (2.6\ people\ /\ submission) = 31,000\ online\ submissions/machine$

will be collected. Per day (with 60 days in the 2-month period), the machine will process approximately 513 online submissions. Focusing on periods of high load, during the busiest time of the day (7-8 pm) where ⅓ of all traffic happens, the peak load the municipal machines will be on average

$$⅓ \cdot 513\ submissions = 171\ submissions/hour$$

which exceeds the maximum allowance of 125 concurrent submissions. While the probability of more than 125 concurrent submissions is low in the span of this hour-long window, without TPS, it's still a likely case to consider. With TPS, the municipal machine can handle a maximum throughput of

$$125\ users \div 31\ minutes\ per\ submission = 242\ submissions\ per\ hour$$

given perfect conditions (i.e. each user logs in exactly when another user logs off). Compared to a design without TPS, other actions in the municipal machines may take precedence over online form submission. Let's say for instance, that 20% of the overhead by chance is used instead to upload paper forms to the local machine. Then, the maximum capacity of concurrent users the machine can support decreases by 20%, and the municipal machine can now handle a maximum throughput of

$$(80\% \cdot 125\ users) \div (31\ minutes\ per\ submission) = 194\ submissions\ per\ hour$$

resulting in a 20% decrease from the throughput using TPS. In this case, the census submission's TPS scheme improves performance metrics for online form collection, an area where availability matters most.

We acknowledge that the limited capabilities of the municipal machines to support online form submission still presents a bottleneck in our system design. Particularly during the end of the census period, the queue wait times to complete an online form may become long during the peak hours each day. Unfortunately, this limitation is hard to overcome by design choices alone and additional system support in the future can help alleviate it.

4.1.2 Data Distribution

During the data upload process, we require the receiver to send ACKs after the census records have been written to stable storage. In addition, encrypted documents take 3x more storage and 3x more time to send. However, we believe that this tradeoff significantly benefits the security of the system.

Each municipal machine is in charge of uploading roughly 100,000 residents' data, who should each have a record of size 800 bytes (2400 bytes once encrypted). Thus, each municipal machine will upload at most 3 * 800 bytes * 100,000 people = 240 MB of data to each user and the national government. Since each municipal machine has a network upload speed of 1000Mbps, it should take a municipal machine 240 MB / 125Mbps = 1.92 seconds of dedicated time to upload the relevant census data to each user. Even with the additional time it takes to transmit due to ACKs and network failures, this upload process should not take more than a couple dedicated seconds on the municipal machine's side. On the national government's side, a similar calculation can be made. A total of (300,000,000 people) * (440 bytes/record) * 3 = 396 GB must be sent to the national government's data centers and a total of (300,000,000 people) * (800 bytes/record) * 3 = 720 GB must be sent to the VMs, which results in a total of 1116 GB of data being sent. Since there are a total of 3000 machines in the national government's data centers, this should take (1116 GB data) / (3000 machines * 10Gbps) = 0.2976 dedicated seconds from each machine to receive. Again, the time it takes to actually receive the data will be slightly higher due to ACKs and network failures, but it will still not take more than a couple of dedicated seconds from each national machine to receive the census data.

Thus, we think the additional overhead from using encryption and our network protocol is justified, as the network is not a bottleneck of our system. The network protocol ensures reliable and accurate delivery of the files between the various components of our system, and the encryption protects the privacy of our citizens, all while still ensuring our census system can still easily meet the April 1st deadline to have the data uploaded and received everywhere it needs to be.

4.1.3 Data Access

Since there are 3000 machines in the national database and it takes 0.07 seconds to read a record, the national database can support ~43,000 census record reads per second. Each municipal machine and its paired VM can similarly perform a total of ~28 record reads per second. As previously mentioned, the network does not present a bottleneck in our system so it will not constrain these speeds.

4.1.4 Data Storage

In this section, we evaluate system longevity and the limitations associated with it with our given hardware constraints.

As calculated in 4.1.2, the national database stores 396 GB of data in the data centers per year. With 3000 physical machines containing 8 TB of storage and assuming the data center resides with 10% of the physical machines, we have

$$10\% \cdot 8\,TB \cdot 3000\,machines/\,(396\,GB) = 6000$$

years worth of storage which is much greater than the intended 70 years. Similarly, the municipal government has plenty of storage capacity in its databases to store the data records. Therefore, storing data records shouldn't be considered a bottleneck to our system.

On the other hand, paper forms stored in the distributed file system of each municipal machine take up a significant amount of storage. For each machine, about (20% * 100,000 persons/machine)/(2.6 households/machine) = 7,690 paper forms that are scanned into PDFs that will be stored. With an average of 2MB per pdf stored and a total storage of 1 TB per municipal machine, the file system can last an estimated

$$1\,TB\ /\ (2\,MB\ \cdot\ 7690\,forms)\ =\ 66$$

years which is less than the proposed specification of 70 years. A clear limiting factor in our system is the municipal storage system. Because the system is required to retain all pdf files, the system can only store approximately 66 years worth of data. While this is close to 70 years, the actual number of years that the system storage can last for is likely much less than 66 due to the high possibility of more users and more data requirements to the system–thus increasing the size of the pdf. By adding more municipal machines, it would greatly help reduce both this problem. Assuming that in the future, the pdf increase to 2.25 MB and we allow for 80,000 persons per machine, the file system can safely last

$$1\,TB\ /\ (2.25\,MB\ \cdot\ (20\%\ \cdot\ 75000\,persons/machine)\ \cdot\ (2.6\,households/machine)\,forms)\ =\ 77$$

years which is a much safer estimate. As a practical solution, after multiple decades of use of the system, we highly recommend the national government to provide more physical machines to the municipalities.

4.2. Failure Resilience

Our system is resilient to various kinds of failures described in the specification. The custom networking protocol discussed in 3.4 ensures that a failure in a machine or the network does not cause the receiving machine to write an incomplete file to disk, and ensures the entire file will be retransmitted when possible. In the case of a network failure between the national and municipal machines, the protocol allows users to still submit census records to municipal machines. These new records will be backlogged and stored on a queue on the municipal machine's file system before being uploaded when the network recovers, which may take up to 3 days. As calculated previously, roughly 640 households submit forms per day which amounts to 640 households * 3 days * 2.6 (people / household) * 800 bytes = 4MB of data to be buffered during this period, which easily can fit on the storage of the municipal machine.

In addition to maintaining the correct internal state, our system also maintains high system availability even in the face of failures on municipal machines or their network. Queries for municipal census data can instead be directed to their paired VMs in the national cloud, as they should have an identical copy of the data to serve the queries. Unfortunately, however, only physical municipal machines can support online form submissions. If an individual municipal machine fails but the rest of that municipality's machines are still functional, the online forms can be completed on the remaining functional machines. If there is only one machine in the municipality and it fails, unfortunately, the users will need to wait for it to come back up.

In the rare event that there is a significant outage in online form submission (i.e. from a network failure, or from greater than half of the municipality's machines failing) during the last week of census

data collection, when census data collection will likely be at its peak, we will extend the census collection period for this municipality by the time the outage lasts. This will not interfere with the municipality's ability to transmit the census data to the relevant parties by the April 1st deadline, since there are just a few hundreds of megabytes of data to transmit on a network bandwidth of 1000 Mbps. Thus, it should take the municipal machines at most a few dedicated minutes to transmit the data to the users. Moreover, this tradeoff partially recovers the availability of our system's census data collection capabilities in the face of this significant outage.

4.3. Use Cases

We now discuss the usage of our system under numerous applicable use cases. We delve into the prioritized user groups and how parts of our system may limit scale.

4.3.1 Government Roles in the System

As C-DOG is a census system intended for use in the realm of government, its main users are the national and municipal governments. Unlike other users–such as the states, election board, and school board–who are simply pushed data from our system, the national and municipal governments must continuously query their respective databases for policy-making purposes.

The national government must essentially access all 300,000,000 records to allocate representation in the national legislature and support national services such as healthcare. Since it only communicates with its own database, queries such as age distribution and population count are precomputed during the data distribution process. This allows for the convenient availability of desired statistics and as a plus, improves performance. However, in the extreme case where the national government must process all 300 million records for an unexpected event (such as another epidemic national emergency), this can be computed in one complete pass which will take approximately 1.94 hours as computed in 3.2.1. This hogs the network, but for less urgent issues similar to redistricting, the queries can be spread out over the course of 2 months. In summary, the range of options for query times supports the availability of national data to implement policies efficiently.

Though, for municipal governments, availability is of utmost importance since they require census data in a timely manner to make swift decisions for multiple services such as police services, fire services, infrastructure maintenance, etc. Therefore, all the census data being stored at the municipal level makes it most available to the municipalities. In addition, as seen in the TPS described in 3.1.2, querying for municipal census data is ranked high. Even in cases where a municipal machine is overburdened or inaccessible, the paired VM has an identical copy as explained in 3.3.

4.3.2 Effect on Researchers

While researchers are allowed to query the database, they are placed at a lower priority tier under municipalities as shown in the TPS. We chose to deprioritize researchers because the census system is clearly meant to play the biggest role for governments. It's unclear what the motivations of researchers could be. For instance, individual researchers could desire census data for privatized business

operations that don't affect Fictlandia. Therefore, our system chose not to allocate space and computation to produce research-like data for the researchers. However, since the data is released after 70 years, the researchers can still benefit after C-DOG is finished to utilize the released data in any shape or form.

4.3.3 Long-Term Changes

While the census system is intended to be upheld for 70 years, Fictlandia will inevitably undergo significant changes. C-DOG can only predict a subset of issues to take care of. Societally, there could exist changes in demographics such as increasing diversity and aging populations. This may require the addition of new questions to the census questionnaire to capture relevant information. The limiting factor in our system is the municipal storage capacity due to the pdf file sizes as discussed in 4.1.4. However, the storage capacity of computers is always increasing and we foresee that by the time storage runs out, there will be other options that can suit the new system requirements. In addition, the percentage of households filling out paper forms taking up pdf storage will likely diminish from the specified 20% due to technological transitions.

While our system upholds security through the use of encryption keys, ultimately, the ever-growing adoption of new technologies could break our security. With limited knowledge today, the best we can do is to monitor the system with hired workers to further improve C-DOG security measures as time goes on.

5. Conclusion

Our proposed system, C-DOG, provides a census structure that improves greatly from the old system. It accomplishes many requirements listed in the specification. Through these requirements, C-DOG maintains its design priorities: *availability* and *security*. Availability is achieved through the TPS that allows for an ordered balance of what users and actions are prioritized given the hardware constraints. In addition, the replication of data in the virtual database allows for a backup in case queries to the local database fail. Security is accomplished through encryption keys that protect the data, and ACLs offer various levels of authorization for different users.

While C-DOG is a major step forward in constructing census systems, ultimately, the system still has finer-grained details to address. In particular, we made the system as available as possible to the government users of the system. For example, the states, municipalities, etc. require swift access to the data in order to perform elections or implement other policy decisions under a time constraint. External researchers, on the other hand, aren't specified any time constraint based on the specification, so we've limited their capabilities based on the specification to just query the system for data. As a result, they have more trouble querying the data since other users and actions are prioritized ahead of them as shown in the TPS. A future improvement for C-DOG could be allocating more computation and storage to perform more in-depth statistics in an organized hierarchical structure dependent on subsets of the data. As a result, researchers can query the system to gain a real-time analysis of the data. Additionally, we omitted accuracy as a consideration, since C-DOG isn't

responsible for the correctness of what users input. However, the addition of ML algorithms can be used to detect fraudulent activity in census data, such as false addresses and other anomalies in the dataset.

6. Author Contributions

The authors of this project all collaborated in designing the overall system, editing this document, and clarifying questions we each had. That said, we each did focus on specific subdomains when writing this report. Roderick contributed to writing the introduction, system overview, and evaluation section (4.1, 4.3), and also designed the diagrams used in this report. Aryan wrote sections 3.1 to 3.4 which focused on how the census system collected, stored, and distributed the census data; in addition, he wrote section 4.2 on failure resilience in our census system. Daniel wrote the section describing the security measures implemented by our census system (3.5) and also wrote the rest of the evaluation section.

7. Acknowledgments

We would like to thank Professor Sollins and Professor Trice especially for their continual feedback, guidance, and support in this project. We received feedback on both communication and technical aspects of our work from each of these instructors, who in turn helped our design grow and develop into the final product it is today. We are extraordinarily appreciative of their support.