Technical Report

UCAM-CL-TR-918
ISSN 1476-2986

Number 918

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Evaluation of decentralized email architecture and social network analysis based on email attachment sharing

Gregory Y. Tsipenyuk

March 2018

This technical report is based on a dissertation submitted August 2017 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Sidney Sussex College.

ISSN 1476-2986

Abstract

Present day email is provided by centralized services running in the cloud. The services transparently connect users behind middleboxes and provide backup, redundancy, and high availability at the expense of user privacy. In present day mobile environments, users can access and modify email from multiple devices with updates reconciled on the central server. Prioritizing updates is difficult and may be undesirable. Moreover, legacy email protocols do not provide optimal email synchronization and access. Recent phenomena of the Internet of Things (IoT) will see the number of interconnected devices grow to 27 billion by 2021. In the first part of my dissertation I am proposing a decentralized email architecture which takes advantage of user's a IoT devices to maintain a complete email history. This addresses the email reconciliation issue and places data under user control. I replace legacy email protocols with a synchronization protocol to achieve eventual consistency of email and optimize bandwidth and energy usage. The architecture is evaluated on a Raspberry Pi computer.

There is an extensive body of research on Social Network Analysis (SNA) based on email archives. Typically, the analyzed network reflects either communication between users or a relationship between the email and the information found in the email's header and the body. This approach discards either all or some email attachments that cannot be converted to text; for instance, images. Yet attachments may use up to 90% of an email archive size. In the second part of my dissertation I suggest extracting the network from email attachments shared between users. I hypothesize that the network extracted from shared email attachments might provide more insight into the social structure of the email archive. I evaluate communication and shared email attachments networks by analyzing common centrality measures and classification and clustering algorithms. I further demonstrate how the analysis of the shared attachments network can be used to optimize the proposed decentralized email architecture.

Acknowledgements

First and foremost, I am very grateful to Vadim and Lisa whose continuous encouragement and support at many levels turned a vague dream into reality and made this dissertation possible.

Many of my family and friends contributed in a myriad ways to this effort and entrusted me with their personal email, which I used in my research. In particular, I owe big gratitude to Natasha for her graciousness, belief in me, and for her incredible patience with my long absences. Special thanks go to Mike and Alla for their amazing hospitality, attention, and support; to Alla for listening and giving critical feedback to some of my ideas; and to Mike for raising the bar on what exemplary study is. My love goes to my mom Inessa, well, for being my mom. I am very grateful to my dear friend Judy for proof-reading my dissertation and inspiring me by her own example.

I want to thank Anil Madhavapeddi for opening my eyes to alternate views until I found the one that piqued my curiosity and ultimately led me to my research topic. I am grateful to my supervisor Jon Crowcroft for letting me to explore different research areas at my own pace, listening to my ideas, however crazy they were, and gently pointing me in the right direction.

Cambridge and Sidney Sussex College became my second home for over four years and gave me many great memories that will stay with me forever. I am especially fond of my good friends Tina and Arathi whose hard work and cheerfulness have been my inspiration. I will always cherish my conversations with them that touched on many philosophical subjects, without any boundaries. I am proud to have them as fellow Alumni.

Last by not least, I would like to thank my examiners, Cecilia Mascolo and Hamed Haddadi for their valuable comments and suggestions that made my thesis better than it was and guided me to meet the standards of research at Cambridge.

Contents

List of Figures

List of Tables

Glossary

AFS andrew file system; is a distributed file system which uses a set of trusted servers to present a homogeneous, location-transparent file name space to all the client workstations. 33

AMS andrew messaging system; is a distributed system project for support of educational and research computing at Carnegie Mellon University. 33

BFS breadth first search is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key') and explores the neighbor nodes first, before moving to the next level neighbors. 24

BFS distributed file system stores files on one or more computers called servers, and makes them accessible to other computers called clients, where they appear as normal file. 25

CAP consistenty, availability, partition theorem; the theorem states that in the presence of a network partition, one has to choose between consistency and availability. 31

CTSS compatible time sharing system; was one of the first time-sharing operating systems. 21

CV coefficient of variation is a standardized measure of dispersion of a probability distribution or frequency distribution. 43

DEM decentralized electronic mail. 35

DHT distributed hash table; is a class of decentralized distributed system that provides a lookup service similar to a hash table: (key, value) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. 22

DMS distributed mailing system. 36

DNS dynamic name system; is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. 27

DoS denial of service attack is a cyber-attack where the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. 36

DTN delay tolerant network is an approach to computer network architecture that seeks to address the technical issues in heterogeneous networks that may lack continuous network connectivity. 25

DVCS distributed version control system; allows many software developers to work on a given project without requiring them to share a common network. 40

EB exabytes = 10^{16} bytes. 69

FOAF is a phrase used to refer to someone that one does not know well, literally, a friend-of-a-friend. 85

FS file system is used to control how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no structure. By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their names is called a "file system". 28

GC garbage collection is a form of automatic memory management. The garbage collector, or just collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program. 49

IM is a type of online chat that offers real-time text transmission over the Internet. 20

IMAP internet message access protocol; is an Internet standard protocol used by e-mail clients to retrieve e-mail messages from a mail server. 21

IoT internet of things; is the inter-networking of physical devices, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data. 3

IP internet protocol; the protocol used to communicate among computers on the internet or the address used for this communication. 30

ISP internet service provider is an organization that provides services accessing and using the Internet. 66

MIME multipurpose internet mail extensions; is an Internet standard that extends the format of email to support text other than ASCII, non-text attachments, message bodies with multiple parts, header information in non-ASCII character sets. 27

MUA mail user agent, an email client or email reader is a computer program in the category of groupware environments used to access and manage a user's email. 27

NAT network address translation; is a method of remapping one IP address space into another by modifying network address information in Internet Protocol. 21

NFS network file system; a file system shared between users on a network and hosted on a centralized server. 34

OSN online social network is an online platform that is used by people to build social networks or social relations with other people who share similar personal or career interests, activities, backgrounds or real-life connections. 23

P2P peer-to-peer; a network topology that has no central control point. 22

PKI public key infrastructure; is a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption. 35

POP post office protocol; is an application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server. 21

SD standard deviation is a measure that is used to quantify the amount of variation or dispersion of a set of data values. 43

SHA1 secure hash algorigthm 1; is a cryptographic hash function designed by the United States National Security Agency. 23

SMTP simple mail transfer protocol; is an Internet standard for electronic mail (email) transmission. 21

SNA social network analysis; is the process of investigating social structures through the use of networks and graph theory. 3

SRS simple random sample; is a subset of a statistical population in which each member of the subset has an equal probability of being chosen. 89

TCP/IP transmission control protocol/internet protocol; provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed and received. 27

UA user agent; email client or email reader. 34

UID unique identifier; identifies email message in a mailbox folder. 31

UIDVALIDITY unique identifier validity value; identifies mailbox folder. 31

UTC coordinated universal time; is the primary time standard by which the world regulates clocks and time. 155

VCS version control system; is the management of changes to documents, computer programs, large web sites, and other collections of information. 49

VM virtual machine; is an emulation of a computer system. 32

XMPP extensible messaging and presence protocol; is a communications protocol for message-oriented middleware based on XML (Extensible Markup Language); it enables the near-real-time exchange of structured yet extensible data between any two or more network entities. 37

ZB zettabytes = 10^{18} bytes. 69

Publications

Publications relating to this thesis:

1. Gregory Tsipenyuk, Jon Crowcroft. My home is my post-office: evaluation of a decentralized email architecture on Internet-of-Things low-end device . In Proceedings of the Second International conference on Internet of Things, Data and Cloud Computing(ICC'17), 2017. DOI: 10.1145/3018896.3018918. ISBN: 978-1-4503-4774-7/17/03. In press by ACMDL. (Chapter 2).

2. Gregory Tsipenyuk, Jon Crowcroft. An email attachment is worth a thousand words, or is it? In Proceeding of the International conference on Internet of Things and Machine Learning (IML'17), 2017. DOI: 10.1145/3109761.3109765. ISBN: 978-1-4503-5243-7/17/10. In press by ACMDL. (Chapter 3).

Chapter 1

Introduction

"For example, of the most interesting questions in technology right now is about centralization vs decentralization. A lot of us got into technology because we believe it can be a decentralizing force that puts more power in people's hands. (The first four words of Facebook's mission have always been "give people the power".) Back in the 1990s and 2000s, most people believed technology would be a decentralizing force.
But today, many people have lost faith in that promise. With the rise of a small number of big tech companies and governments using technology to watch their citizens many people now believe technology only centralizes power rather than decentralizes it."

–Mark Zuckerberg. Facebook Founder and CEO. New Year Resolution, January 2018[163]

Computer-scientist, philosopher and musician Jaron Lanier in his book "Who Owns The Future" [81] talks about our addiction to Siren Servers. A Siren Server is a powerful computational resource that out-computes everyone else on the network and grants its owners a guaranteed path to unbounded success. Lanier alludes to Google, Facebook and other Internet companies. He believes that the middle class is increasingly disenfranchised from the online economy, yet individuals offer an amazing amount of value and the lion's share of wealth now flows to those who aggregate and route those offerings, rather than those who provide the "raw material". We expect online service to be given for free, but nothing is free and we pay for it by acquiescence to being spied on. Lanier believes that this kind of economy is not sustainable and if it continues then we are setting ourselves up for more unemployment or social backlash. Capitalism only works if there are enough successful people to be the customers. Consequently, Lanier says that we should seek a future where more people will do well, without losing liberty, even as technology gets much, much better. The way to accomplish it is to pay people for information gleaned from them if that information turns out to be profitable.

According to a Cisco forecast[1], emergence in recent years of IoT as a new computing paradigm will see the number of interconnected devices to grow to over 27 billion by 2021. The vast amount of data which are going to be generated by these devices presents both opportunity and challenge for producers and consumers of the data and mirrors concerns raised in Lanier's "Who Owns The Future". Policy makers see the importance

[1]`https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html`

of consumer protection against Siren Servers in the age of the digital economy. General Data Protection Regulation (GDPR)[2] in the EU seeks to strengthen and unify the data protection for all citizens within the EU. The GDPR aims primarily to give control back to citizens and residents over their personal data and to simplify the regulatory environment for international business by unifying the regulation within the EU[3]. In the USA a similar legislation, the Consumer Privacy Bill of Rights[4], was introduced in 2012. The bill is the framework for protecting privacy and promoting innovation in the global economy. Crabtree et al. in [44] have the view that these policies are not just about consumer protection but are also about enabling a new kind of economic actor who is actively participating in the digital and emerging data economy. From this perspective, the user becomes an active data trader. Researchers follow suit in consumer's protection policies with the framework, which ensures user's privacy and share in the profits from the online economy. For instance, personal Databox by Haddadi et al. in [73], offers a technical platform which enables users to manage, collect, and consume personal data under direct control of the individual whose data it holds. The Databox device is located at the individual's home and collects data from physical sensors, Internet, or social media "data sources". The Databox is the gateway to individual or individuals "data sources". Crabtree et al. in [43] extend the Databox model to the IoT environment. In this setup, data processing is moved from the cloud to the edge of the network to enable local control and minimize distribution of personal data and any threat to privacy.

I conclude from the above that individuals using centralized Internet services, like Email or Online Social Network, are excluded from benefiting in the digital economy and risk having their privacy violated. On the other hand, policy makers, understanding this social dis-balance, put forth legislation to protect a user's privacy and encourage participation in profit sharing in the digital economy. Likewise, researchers follow suit with platforms to provide technical means for protecting user's data and turning it into monetary or barter value. Therefore, in the thesis of this dissertation I am looking at two research questions. First, I hypothesize that the recent phenomena of IoT presents an opportunity for email decentralization in the way that takes user's data out of Siren Servers influence and puts it back under the user's control. I propose a high-level email architecture and evaluate the feasibility of this architecture on a resource-constrained IoT device. Second, I suggest that the data, which is the driving force behind the digital economy, should be analyzed to provide input into the design process of the technical platforms. Consequently, I hypothesize that the social network extracted from email attachments, which constitute the bulk of the data in email messages, may provide valuable insight for architecture optimization. I evaluate graph and node-level metrics for both attachments and conventional communication networks, demonstrate that what we can learn from the attachments network is complementary to the communication network, and show how we can use this knowledge to optimize the proposed decentralized email architecture. While email enjoys high popularity, and still has the highest number of user accounts in 2017 at 4.9 billion versus 4.8 billion of social network accounts, Neely [112], and 3.5 billion of Instant Messaging (IM) accounts, Radicati [121], other means of communication are catching up with email. Consequently, I demonstrate how this methodology can be applied to the structural analysis of the online social network.

[2]https://www.eugdpr.org/key-changes.html
[3]https://en.wikipedia.org/wiki/General_Data_Protection_Regulation
[4]https://epic.org/privacy/white_house_consumer_privacy_.html

Email has over a half century history and was quite likely the first Internet killer application and first distributed social network. Yet, conceptually it is simple and is supported by two main functions. First is email transmission, which is based on Simple Mail Transfer Protocol (SMTP) [87]. Second is email retrieval, which is based on Internet Message Access Protocol (IMAP) [46]. Original definitions of SMTP and IMAP standards go back to 1982 and 1994 respectively. IMAP was preceded by a less advanced Post Office Protocol (POP) dating back to 1988. But the first implementation of email goes back even farther and predates the Internet. One of the earliest email application is attributed to MITs Compatible Time Sharing System (CTSS) [3]. In this early incarnation of email, a message is added to the designated user's mailbox file, which is essentially a file append operation. In 1971 Ray Tomlinson [5] sent the first networked email. This also gave birth to now ubiquitous "@" sign in the email address as the way to separate the destination's user name or the mailbox and the host name or the domain. Popularized by AOL's "You've got mail" [1], centralized email service as we know it today evolved in mid-1990s with Hotmail being one of the earliest providers [6]. Centralized servers provide valuable features to users like the ability to send a message to a user who is not on-line (store-and-forward), transparent connection for users behind a firewall and Network Address Translator (NAT), redundancy, availability, and backup.

Yet email is not without flaws. As one would expect from an application seen from the dawn of the Internet, there are number of protocol extensions defined over the years to keep up with evolving technologies and demands from users. IMAP protocol alone has over sixty extensions[5]. This continuous patching makes protocols more complex and not necessarily more efficient. Moreover, the email server and the client implementations do not support every extension.

Some studies, Castro et al. [36, p2] and Grbovic et al. [68, p1], show that email users usually do not delete their email, keep them in their Inbox, and extensively use an email search capability [45]. Castro et al. also find that 89% of users delete some emails without reading [36, p3]. Moreover, we now access our emails from multiple devices and predominantly from our smartphones and tablets and to a lesser extent from laptops or desktops. Consequently, important messages can be unintentionally deleted or misfiled and mailbox synchronization may result in changes made on some client devices being lost. While some government agencies, in order to comply with the law, may keep all, even deleted emails, it is not natively handled by IMAP protocol.

As part of my research I analyzed private email archives of friends and family and Enron's energy trading company email corpus (Section 1.1). I found that attachments use 91% and 81% of disk space in Enron and private archives respectively. Moreover, duplicate attachments use 34% and 27% of disk space in Enron and private archive respectively (Table 3.1). This means that if single storage is not implemented for an email message then disk space and energy is not used in the most efficient way. Similarly, bandwidth and energy might not be used in the most efficient way if attachments are redundantly synchronized between a client and a server. This highlights the importance of the data-driven design in the age of the digital economy. The data is also expensive in terms of the energy cost that is required to run the data centers, which consume 3% of the global energy and contribute 2% to green house emissions. Our email is not free and harmless!

Indeed, email is not free. There is an intangible cost involved to users; namely invasion of privacy. Email providers are businesses which have to recover their infrastructure costs

[5]https://www.imapwiki.org/ImapRFCList

and make a profit for investors. This is accomplished by data-mining user email archives and selling the information to advertisers.

Indeed, email is not harmless. Or to be more exact, the fact that email is centralized makes it an attractive target for recreational or criminal hackers, government surveillance, and even as a way to destabilize the democratic process. Since Snowden's revelations in 2013 [11] about the National Security Agency surveillance Prism program, there has been many email-hacking news stories. Yahoo acknowledged that over one billion accounts were stolen in 2013 and over 500 million accounts were stolen in 2014; Goel and Perlroth [66]. In 2016, 272 million accounts from Mail.ru, Gmail, Yahoo, and Microsoft were stolen, Auchard [20]. Moreover, emails have been hacked possibly with an intent to influence elections in the US, Harding [76], and France, Breeden et al. [33]. And then there is Clinton email's controversy [15] about improper use of her personal email server for government correspondence, which highlights the fragility of the centralized email server.

It is natural then that email presents many opportunities for research. It is interesting how email research evolved. Initial research prior to 2000 revolved around areas that contributed towards centralized services. Later research was driven away from centralized architecture by the realization of the flaws inherent in the centralized system and success of Peer-to-Peer (P2P) file-sharing applications. P2P provides many benefits such as robust wide-area routing architecture, efficient search of data items, selection of nearby peers, redundant storage, permanence, hierarchical naming, trust and authentication, anonymity, massive scalability, and fault tolerance, Lua et al. [96, p1]. Conceptually, the solution to email decentralization in P2P network is to replicate and distribute email messages between participating user computers. Placement and routing of a message is typically accomplished via Distributed Hash Table (DHT)[6]. In spite of the many benefits that P2P architecture may have provided, it did not resonate with users. Perhaps entrusting their email, even encrypted, to distributed unknown users, was no more attractive than having it known by some centralized provider. In addition, the location and consequently the meaning of data ownership, is probably more vague in P2P case as well. Moreover, P2P email architecture generally keeps SMTP and IMAP protocols, consequently inheriting their flaws, for instance inefficient email synchronization.

Does the recent phenomena of IoT present an opportunity not just for smart homes but for email decentralization as well? I hypothesize and demonstrate in this thesis that the answer to this question is yes. The"things" in IoT can not only order milk, drive cars, or control the lighting system but they are also capable, globally-addressable, computers connected to the Internet. Essentially a user can have her own P2P network consisting of smartphones, tablets, and all IoT devices which are part of the things that she owns. This P2P network can provide an access path and storage redundancy, availability, privacy, and reduce hacking vulnerability.

IoT and digital economy, in particular, are about the data and there is a lot of it being moved around. Cisco forecasts that annual global Internet traffic will reach 3.3 ZB by 2021, which is almost a three fold increase from 2016[7]. Consequently, data analysis or data-driven design should be considered as part of the system architecture. In this respect, email is not an exception, and email archives present a valuable resource for SNA research field. For instance, analysis can be used to discover relations and hier-

[6]https://en.wikipedia.org/wiki/Distributed_hash_table

[7]https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

archies, the most influential people, hidden groups, financial fraud, or crisis prediction. Conventionally, a communication network is extracted from the structured email headers to run SNA. A communication network provides the direction and frequency of communication for a group of users who are the subject of the research. This type of network ignores email attachments which may constitute the bulk of the data in the email message. Consequently, extracting the network from email attachments shared between users may provide more insight into the information interaction within the network and complement the analysis of the conventional communication network. Moreover, the analysis of the shared email attachments network can be used to optimize proposed decentralized email architecture by minimizing the number of replicas necessary for the email messages backup and availability. This approach takes advantage of the social relationship between users and creates a backup group of closely related people who use each other devices for the backup. Because the users within this group interact more with each other and share data, these data does not have to be redundantly replicated, consequently reducing the cost of energy. In addition, this methodology can be extended to the Online Social Networks (OSN) analysis, demonstrating general applicability of this approach.

1.1 Experimental datasets

In my analysis I use three datasets:

- Private email archives were provided to me by my family members and friends. I collected 29 email archives with the overall size of 68.2 GB. I wrote two IMAP client applications for MAC OS X and Windows to download and anonymize the data so that no private user information is collected. To protect the user's privacy, email's message-body and attachments are removed and the headers are hashed. The extracted email data has the headers containing *From, To, Cc, Bcc, Date, Subject, Mailbox, Message-ID, Inreply-To* fields, email's body size, and compressed size. *From, To, Cc*, and *Bcc* fields are converted to their Secure Hash Algorithm 1 (SHA1) [58] hashes. Since I know the email address of the core 29 people, I can infer the original email address from the hash but I can not infer other email addresses present in the header. The *Subject* is converted to its SHA1 hash. The Mailbox is mapped to a unique number, with some default mailboxes like *Inbox, Sent, Deleted*, etc. having a persistent number. The email's body, when applicable, is parsed into MIME parts, with each part containing the *Content-Type* header, number of lines in the header, size of the header, compressed size of the header, body size, and body compressed size. If the body is an attachment, then the attachment's SHA1 hash is provided. To provide better visual representation in tables and figures, I mapped email addresses of the core 29 users to a name in the format *userN*, where *N* is a random number. The Windows application is written in C# and MAC OS X application in OCaml as part of IMAP/SMTP suite. Both programs are available on GitHub[8]. The email messages are downloaded and anonymized with the provided software by users who then deliver the final product of the email extraction and anonymizaton to me. I explained to users how the data are extracted and anonymized, how the data are going to be analyzed, and what information about users can be inferred from the data.

[8]`https://github.com/gregtatcam/email_proc`,`https://github.com/gregtatcam/imaplet-lwt`

- The Enron email corpus was released by the Federal Energy Regulatory Commission during the investigation into Enron's collapse [4] and is publicly available. Appendix A discusses the data processing applied to the Enron email corpus.

- Flickr is an on-line photo management and sharing application[9]. I obtain the Flickr sample dataset by conducting Breadth First Search (BFS) of favorites photos. I start the search with ten seed users from National University of Singapore Flickr dataset[10]. In each step of the crawl I retrieve the list of the user's favorite photos, or photos of which she is the fan, and add the owners of the photos to the list of users to visit. The search is stopped when the number of fans exceeds two million. Overall, the crawled dataset contains 2 000 177 users.

1.2 Thesis contributions

The contributions of this thesis can be summarized as follows:

- I present a high-level decentralized email architecture which maintains a full email history of changes in a Revision Control System-like back-end. The architecture replaces legacy IMAP and SMTP protocols with a synchronization protocol based on Merkle hash tree, Merkle [105].

- I present detailed evaluation of the latency, CPU, bandwidth, energy, memory, and disk usage for various types of the email storage on a Raspberry Pi and compare them to IMAP Dovecot server. I also analyze the energy usage of the decentralized architecture and compare it to the conventional centralized architecture. I demonstrate that the proposed architecture is feasible on a resource-constrained device like a Raspberry Pi and that it performs at least as well as a conventional IMAP server.

- I extract communication and shared attachments networks from private and publicly available email datasets, calculate common centrality measures and run k-nearest neighbor classification and k-means clustering algorithms. I further demonstrate that the shared attachments network provides additional insights into the social relations within the network. Moreover, I demonstrate how the same analysis can be applied to the photo-sharing Flickr social network to discover groups of shared interests.

- I define the energy cost saving model. Saving is realized through the optimized backup strategy, which minimizes the replication cost. The strategy takes into consideration the social relationships between the users. The model is evaluated on the experimental datasets used in this research and demonstrates that there is energy cost savings in the groups constructed from the social information contained in the datasets as opposed to the randomly generated groups.

[9]`https://www.flickr.com/about`
[10]`http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm`

1.3 Thesis outline

Chapter 2 briefly reviews the history of email, email advantages, protocol flaws and opportunities presented by the IoT phenomena. I review the background research that led to the creation of centralized email services and consequent research that looked at how to move away from centralized architecture by utilizing P2P network. In addition, I review the literature on Distributed File System (BFS) and Delay Tolerant Network (DTN) since some properties of these systems mirror the decentralized email architecture. High-level decentralized email architecture is proposed and detailed evaluation demonstrates that it is feasible. Availability of the system is analyzed. The chapter concludes by presenting cost comparison analysis of the proposed and centralized architecture.

Chapter 3 reviews conventional communication network SNA and related background work on one-mode projection graphs, tie-strength definition, structure inference in the social network, and use of SNA in system design. General network statistics, centrality measures, k-nearest neighbor, and k-means clustering is analyzed for communication and shared attachments networks of the email datasets, and k-means clustering is used for the structural analysis of the Flickr social network. The chapter concludes by demonstrating application of the shared attachments and OSN SNA to the decentralized email architecture optimization.

Chapter 4 concludes the thesis by presenting the summary of the thesis contributions and future work.

Chapter 2

Evaluation of decentralized email architecture on Internet of Things resource-constrained device

2.1 Background

2.1.1 Protocols and concepts

In this section I provide an explanation of protocols and concepts used in present day centralized email architecture and other protocols, concepts, and evolving technologies that have to do with email decentralization.

Figure 2.1[1] illustrates a high-level centralized email architecture. The figure demonstrates roles played by email and supporting protocols. The sequence of events is as follows. Alice composes an email in her Mail User Agent (MUA), which is the email client. MUA sends the email to Alice's SMTP server. The server looks up Bob's email server public address record in the Dynamic Name System (DNS) server and sends the email to this address. Bob, at a later time connects to his POP or IMAP email server and retrieves the email. In the thesis I focus on SMTP and IMAP email protocols and Internet Message Format and Multipurpose Internet Mail Extensions (MIME), Freed and Borenstein [61], standards, which define the format of the email message.

Simple Mail Transfer Protocol (SMTP, RFC 5321) is an outgoing server. When an email client needs to send an email, it connects to the defined user's SMTP server. Typically, the connection is established over secure Transmission Control Protocol/Internet Protocol (TCP/IP) channel and the user is authenticated. The client could be another SMTP server, or so-called relay server. The relay server is needed when an email is sent to a domain, different from the domain of the outgoing server. For instance, an email is sent from Google to a Yahoo account. SMTP transports the message and does not concern itself with the message content, only with its envelope; i.e., the sender and the recipient.

Internet Message Access Protocol (IMAP, RFC 3501) is an incoming server. When a user's client needs to read an email, it connects to the IMAP server, typically over secure TCP/IP connection, and authenticates itself. Once connected, the client can retrieve, search, or make limited modifications to messages and mailboxes in which

[1]Reproduced from `https://en.wikipedia.org/wiki/Email`

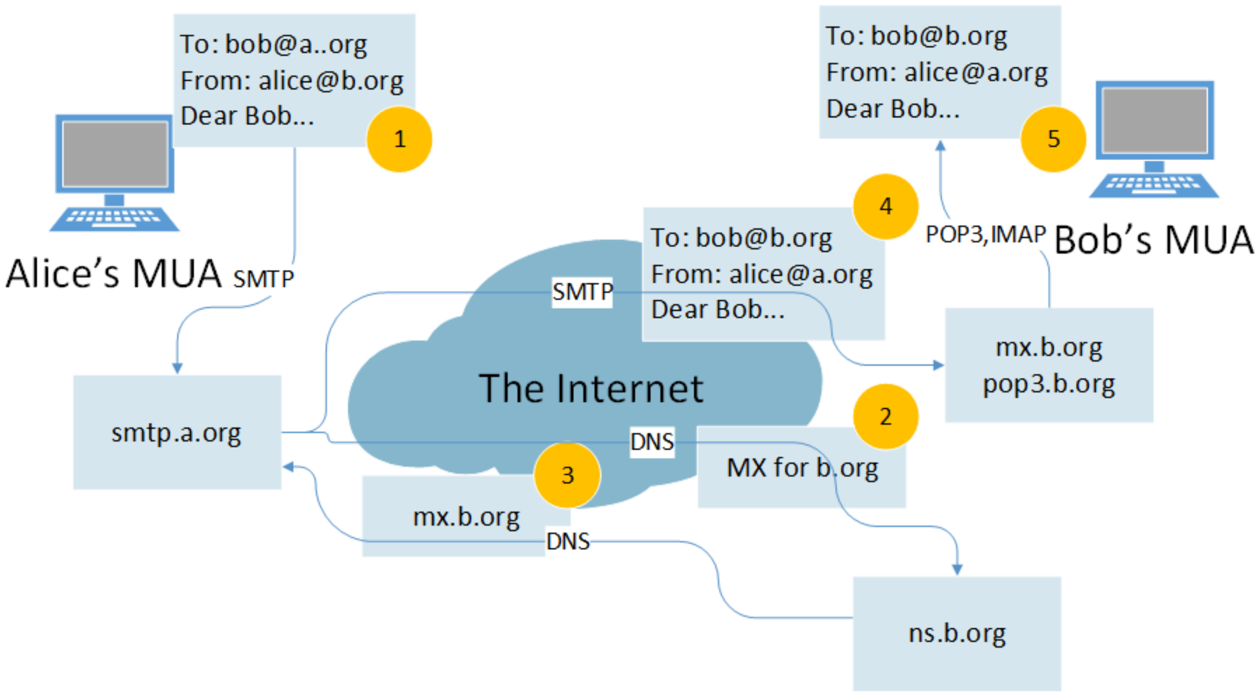


Figure 2.1: High-level centralized email architecture.

messages are contained. Some clients maintain connection to the server in order to receive notifications about new and updated messages. Unlike the SMTP server, which is fairly simple and has fewer commands, IMAP is a complex state machine with a sophisticated set of commands and many protocol extensions. IMAP supports multiple clients with a somewhat inefficient email synchronization mechanism via folder and message unique ids. In a way IMAP is a remote File System (FS) which allows manipulation of logical folders and files - email messages.

Internet Message Format (RFC 5322) and Multipurpose Internet Mail Extensions (MIME, RFC 2045) is a set of standards that control the email message format. RFC 5322 defines the overall format of the email message as consisting of the structured header fields and unstructured message body. The only required header fields in an email message are *Date*, which indicates the date and time of message completion, and *From* field, which identifies the author of the message. Typically, there are at least destination address fields *To, Cc, Bcc*, which identify primary, carbon copy, and blind carbon copy recipient(s), respectively; *Subject* field, which identifies a topic of the message; *Message-ID*, which is a global unique identifier of the message. Many other standard and custom defined headers are possible. MIME standards extend RFC 5322 to deal with multi-part message bodies, characters in the email message other than US-ASCII, sets of different formats for non-textual message bodies, and textual header information in character set other than US-ASCII. My thesis is concerned with the multi-part message body RFC 2045 only. This RFC deals with the email message body consisting of other embedded messages and attachments.

Peer-To-Peer (P2P, RFC 5694) is an overlay network built on top of another network, for instance TCP/IP. A pure P2P system does not support any hierarchy or centralized control and is self-organizing. Nodes in P2P system share their resources to provide certain services. Nodes in P2P play a dual role in that they both provide services and request

Figure 2.2: Artist's impression of IoT.

services from other nodes. There are hybrid P2P systems where a centralized server can provide some services. For instance, in a file sharing P2P system, the centralized server can contain the list of files and nodes containing them. Nodes joining a P2P system have to discover other peers and also have to be authorized and authenticated. Some of these functions could also be handled by a centralized server, which is called a bootstrap server. Some typical functions provided by P2P systems are data indexing, data storage, computation, and message transport. P2P can be classified into unstructured where the search for information is done via flooding and structured, where the search is done via routing, for instance with DHT, though there are different interpretations of structured and unstructured classification by different authors. P2P systems have to protect themselves against many types of attacks, for instance Sybil attack, Douceur [52], where an attacker subverts peer reputation system by forging a large number of false identities under attacker control. They also have to deal with a high churn rate when peers frequently join and leave the system.

Revision Control System (RCS) is typically used in the software or document management system to control revision of files. RCS can be standalone or distributed. It could maintain revisions as plain files or in the database. There are different ways in which changes to the file are maintained. It could be implemented as tracking changes to individual files or as snapshots. Regardless of the specific implementation of RCS, they all generally have the same features of tracking document changes and ability to retrieve any specified document revision. In my thesis, I am interested in RCS maintaining files via content addressed storage and snapshots via Merkle tree. In this system email messages can be stored as MIME parts, providing efficient single storage of attachments and efficient synchronization of distributed archives.

Merkle Hash Tree is a tree in which every leaf node is labeled with the hash of a data block and every non-leaf node is labeled with the cryptographic hash of the labels of its child nodes. Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains [105]. Hash trees

can be used to verify any kind of data stored, handled and transferred in and between computers. They can help ensure that data blocks received from other peers in a P2P network are received undamaged and unaltered, and even to check that the other peers do not lie and send fake blocks. Hash trees are used in distributed revision control systems[2].
Internet Of Things (IoT) is the inter-networking of devices embedded with sensors, electronics, software, actuators and network connectivity which enables these devices to collect and exchange data[3]. An IoT device does not just collect data but can also perform some action. The device could be just a sensor or a capable mini-computer and may penetrate every aspect of our lives at home, at work, and in between, including our health as demonstrated by the artist's impression in Figure 2.2[4]. It is estimated that by 2021 there will be over 27 billion wireless connected IoT devices.
Internet Protocol version 6 (IPv6) is the most recent version of the Internet Protocol, the communications protocol that provides an identification and location system for computers on networks and routes traffic across the Internet. IPv6 was developed by the Internet Engineering Task Force to deal with the long-anticipated problem of IPv4 address exhaustion. IPv6 is intended to replace IPv4. The design of IPv6 intended to re-emphasize the end-to-end principle of network design that was originally conceived during the establishment of the early Internet. In this approach each device on the network has a unique address globally reachable directly from any other location on the Internet[5].

2.1.2 Overview

Email predated the Internet and has contributed to its evolution by becoming the first killer application, Biersdorfer and Pogue [25, p251]. By 1973, email constituted 75% or ARPANet traffic [2]. It still enjoys high popularity and is ranked as the top Internet activity [32, 14, 101]. The email application started as a simple file manipulation in MIT's CTSS, with messages being appended to the user mailbox file on disk, Vleck [148]. With the evolution of the Internet and introduction of middle boxes, email moved to centralized services running in the cloud. Centralized architecture is required for following reasons:

- Email clients are by and large located behind a firewall or NAT and therefore do not have globally-accessible addresses. Central service allows a client to connect to the server from anywhere in the world via the domain name, which is resolved by DNS to public Internet Protocol (IP) address.
- The sender and the recipient are generally not on-line at the same time. The message is sent to the recipient's email server where it can be retrieved at a later time. This store-and-forward property of the email is important for devices, which may have intermittent connection to the Internet.
- Central service provides high availability via redundancy and replication.
- Email archives are backed up.

While the services are free to users, there are associated intangible costs:

[2]https://en.wikipedia.org/wiki/Merkle_tree
[3]https://en.wikipedia.org/wiki/Internet_of_things
[4]Reproduced from https://en.wikipedia.org/wiki/Internet_of_things
[5]https://en.wikipedia.org/wiki/IPv6

- Privacy. To recover high costs associated with data center maintenance, an email provider data-mines user email and sells the information to advertisers.

- Hacking. The centralized nature of email is a high-value target for recreational, criminal, and government hacking, Berghel [24], Kopstein [89], Madden [97], [8].

According to Edwards et al. [59, p6] in Bayou, electronic mail is often considered to be the "classical" asynchronous collaborative application. This type of network-shared data system according to Brewers CAP theorem [34] is characterized by high availability and tolerance to network partitions with an eventually consistent database; i.e., mailbox in case of the email. These properties are inherent in delay tolerant email store-and-forward architecture. Generally, an email client, except for the web-based, has a mailbox replica with the centralized server maintaining a full version of the mailbox. Synchronization of the replica with the server mailbox is handled by IMAP, Crispin [46], and to some extent by SMTP, Klensin [87]. SMTP protocol handles outgoing messages to other email accounts, indirectly updating their INBOX database. IMAP protocol handles incoming messages, updates email metadata on the server, and synchronizes the replica on the client. Primary keys used in email synchronization are unique (within a mailbox) message id (UID) and unique mailbox id (UIDVALIDITY), both assigned by the server. Consequently, the client needs to run additional queries after an update to retrieve UID. For instance, after copying a message to another mailbox the client has to search the mailbox for the copied message UID. In addition, the protocol does not guarantee UID to be consistent between sessions. Changes made to the same mailbox by multiple clients can result in communication overhead between the client and the server in order to resolve inconsistencies; for instance, one client deleting a mailbox and another renaming the same mailbox. A number of extensions have been added to address protocol limitations. IMAP extensions like MODSEQ, Melnikov et al. [104], or IDLE, Leibal [94], provide more granular synchronization and ability to receive unsolicited email update notifications. Either way, the server can only communicate the aggregate updates from all clients since the last synchronization. Because SMTP and IMAP are independent protocols and the corresponding servers are not necessarily physically co-located there may be other network overhead. The message could be sent twice - to the SMTP server for relay and to the IMAP server Sent Messages mailbox, though this could be remedied (if implemented) by BURL SMTP extension [114].

Schmandt and Marti [131, p25] pointed out in 2005 that mobile email usage is growing fast with increasingly heterogeneous multi-device access to email. Since then the Internet access has shifted towards mobile devices, with the number of mobile users exceeding the number of desktop users in 2014, Chaffey [39], and mobile email access overtaking the desktop in 2011 [10]. This transition to mobile computing might play a role in user email actions. Castro et al. in [36, p3] show that 89.5% of email delete actions are delete-without-read; i.e., users delete the email without even opening it, let alone reading. Authors suggest that this phenomena could be explained by the increased number of machine-generated email, which accounts for 90% of non-spam Web email, Grbovic et al. [68, p2]. It is also possible that as users tend to check their email on mobile devices while busy with other activities, users have less patience to read the entire message and delete it based on cues such as the message subject or preview, Schmandt and Marti [131]. Another interesting observation in Whittaker et al. [152] is that users prefer scroll and search rather than folder-access when they need to re-find an email. This might be partially explained by the limited UI capabilities on mobile phones where streamlined sort and search are more efficient. Nevertheless, the folder access accounts for 12% of overall access.

In the Introduction I discussed the privacy problem inherent to the centralized service. In addition, I see the following issues affecting email as it is accessed from multiple intermittently connected devices. First, changes from multiple clients eventually have to be resolved on the server with the most recent update overwriting the previous ones. This maybe undesirable or simply not what the user wants. Second, a user may unintentionally delete an important message or file a message to the wrong or obscure folder. To address these issues it is not sufficient to have the latest email state. As Brewer notes "The state is less useful than the history, from which the system can deduce which operations actually violated invariants and what results were externalized, including the responses sent to the user" and "The best way to track the history of operations on both sides is to use version vectors, which capture the causal dependencies among operations" [35, p5,p4].

Recent phenomena of IoT will see the number of interconnected devices grow to 27 billion by 2021. A device could be a home router, an electricity monitor, or an entertainment system. While a resource limited, some of these devices are comparable in the hardware configuration to an average smartphone, and might have a globally accessible address. For instance, the latest release of the popular Raspberry Pi features 64 bit Quad Core 1.2 GHz ARM Cortex A53, 1 GB 900 MHZ RAM, 2.4 GHz 802.11, Bluetooth, 10/100 Ethernet, 4xUSB 2.0, and microSD storage up to 200 GB. To put things in perspective, the Andrew message system defines as high-function computers that have 2-4 MB RAM and 40-70 MB hard drive, Rosenberg et al. [124, p1]. In all fairness, this was almost 30 years ago. Even though email has not changed much conceptually since then, the environment has. IoT along with challenges presents opportunities to change email back to the decentralized architecture as it was at its inception.

The contribution of this chapter are as following:

- I am proposing a high-level email architecture where the complete revision history of the email is stored on user devices whether it is a home router, a Virtual Machine (VM) in the cloud, or a mobile device, forming a cluster of user owned devices. Moving the data to user devices puts the user in control of its privacy. The user can sell the data directly to advertisers, let the cloud provider data-mine her email in exchange for the VM storage, buy VM storage, or just use her own IoT device with globally accessible address. Even if the data are released to a third party, confidentiality can still be maintained via homomorphic encryption like in the Mylar system, Popa et al. [118]. Replication between multiple devices provides redundancy, availability, and backup even if the cloud VM is not used. Maintaining the full history of the email provides for eventual consistency of divergent replicas in disconnected devices and addresses possible inconsistency caused by email access from multiple devices and deleted or misplaced messages due to erroneous user actions. The interconnected network of user clusters can use a synchronization protocol to replicate email between user devices instead of the IMAP protocol. Synchronization protocol only replicates new files, preserving the bandwidth and the energy by not copying duplicate attachments (Section 2.2).

- I present a detailed evaluation of the email architecture on Raspberry Pi computer (Sections 2.3, 2.4.1, 2.4.2, 2.4.3).

- The evaluation shows that the approach is both feasible and affordable (Section 2.4.4).
- Distributed architecture evolves out of 1) taking a modern view of what email architecture requirements are: including eventual consistency for synchronization, and modern approaches (i.e. CAP) to consistency; 2) the advent of IoT, solving the reachability and ubiquity problem of availability of P2P (Section 2.2).
- I provide limitations of the methodology and evaluation (Section 2.5).
- I analyze availability of the system (Section 2.6).
- I compare the energy cost of the proposed and centralized architecture (Section 2.7).

2.1.3 Related work

The time-line for email research can be roughly divided into two eras separated around year 2000. The year is significant in that from 1999 to 2001 Napster[6], Gnutella[7], and BitTorrent[8] P2P file-sharing services were launched. P2P leverages the computing resources of cooperating users to achieve scalability and organic growth, Rodrigues and Druschel [123]. Success of the early P2P systems gained research attention in other areas with the email being one of them. Consequently, email research after 2000 is largely focused on P2P architecture. I first look at the early research, which generally contributes towards the evolution of conventional centralized email service.

2.1.3.1 Towards centralized service

Grapevine, one of an earlier distributed and replicated email systems, has a user's Inbox distributed over multiple servers, Birrell et al. [28]. Email is delivered to the nearest available message server, which in turn relays the email to a server containing user's Inbox. Multiple Inboxes are made transparent to the user via registration service. Grapevine's naming schema for services and users inspired the DNS service, Partridge [115]. Grapevine supports replication of the delivery path rather than the messages, considering the extra complexity as not worth the added availability. Consequently, the registration services data are replicated but in a way that trades atomic update over increased availability. Grapevine's contribution to the centralized architecture is replication of the delivery path. Essentially it implements POP protocol with the message being discarded once it has been retrieved and the client being responsible for the message backup. The expectation is that the message retrieval latency is small, consequently making the probability of the server failure small as well. This approach may indeed work well within the single organization but not in a global centralized server.

Andrew Messaging System (AMS) in Rosenberg et al. [124], features a DFS, or Andrew File System (AFS)[9], emulated as monolithic Unix FS. AFS uses weak consistency with read and write operations on an open file directed only to the locally cached copy. The message database in AMS is structured as directories with files, with the message stored

[6]https://en.wikipedia.org/wiki/Napster
[7]https://en.wikipedia.org/wiki/Gnutella
[8]https://en.wikipedia.org/wiki/BitTorrent
[9]https://en.wikipedia.org/wiki/Andrew_File_System

in a single file. A directory can have messages and other subdirectories. AMS focuses on reliability, which is ensured by AFS volume replication to read-only cloned copies. The key AMS contribution to centralized architecture is its DFS with the file replication. This DFS might be a precursor for Datacenter DFS in the present day centralized server, consequently contributing towards its weaknesses like single point of failure, lack of organic growth, and high Datacenter costs.

Porcupine email server in Saito et al. [127], consists of a cluster of nodes. Unlike from large scale email servers, there is no role separation between the nodes. That is, each node runs SMTP, POP, and IMAP sessions. Users are distributed between nodes via hashed user name. User's mailboxes are also distributed between the nodes. Similar to AMS, Porcupine provides a single-file-system view via Network File System (NFS) gateway. Porcupine contributes to the centralized architecture with its clustering mechanism which essentially is replication of the delivery path like in Grapevine. But it does not replicate messages making it less reliable in case of a failure.

NinjaEmail in von Behren et al. [149], uses Ninja cluster architecture for high performance local-area email service and OceanStore storage management to connect multiple clusters. Distributed Data Store in the form of DHT replicates data over the subset of cluster nodes. OceanStore handles replication between the clusters and conflict resolution to provide eventual consistency for modified data, and migrates the data close to clients to improve the latency. One distinct feature of NinjaEmail is that sending a message simply appends the message to the user's Inbox in the OceanStore. With its local clustering and geographically distributed replicated storage, NinjaEmail is a perfect example of centralized architecture. As such it also contributes to its weaknesses described above.

In early email research, we see the evolution of features like DFS and clustering to provide high-level of availability and redundancy.

2.1.3.2 Decentralizing with P2P

As mentioned above, recent email research focuses on P2P architecture, which takes advantage of participating peer's resources to address shortcomings of centralized system such as scalability, availability, single point of failure, and privacy. In Kangasharju et al. [84], the system node in DHT provides persistence for messages in transit from the sender to the receiver. Nodes maintain three types of objects: email address certificates, email message bodies, and Inboxes. Inbox stores notifications to unread messages. User Agent (UA), the client's application, which does not necessarily run on the system node, is responsible for the email replication between peers closest to the calculated object's key. UA reads messages from all user's Inboxes to reconcile inconsistencies due to multiple nodes maintaining the instances of the Inbox. Messages are not persistent and deleted after reading.

Bayou in Edwards et al. [59], is a replicated weakly-consistent storage system. BXMH is the email client with the file handling implemented via Bayou relational database. A BXMH user runs a Bayou server on each machine where the mail will be read. Changes to the mail can be made while disconnected. When the machine is reconnected, the email database differences on each server are reconciled. Bayou introduces Timewarp, a higher level toolkit, which provides the versioning functionality. Timewarp is used for collaboration so that participants can view different revisions and make changes to an artifact in any point of its history. Formally, Bayou does not term itself a P2P architecture perhaps because it was released in 1997, two years prior to Napster. It does have properties of P2P in that each email client runs Bayou server, the updates are eventually propagated to all replicas, and the user can choose what replica she is connected to. In this way users are contributing their computing resources to the good of the overall system.

In Zhao et al. [159] hybrid P2P system, the super node either plays the role of the centralized server (pseudo-cooperative schema); provides lookup services, assigns replication and sends tasks to regular nodes, and maintains temporary information about replicated messages (simple-cooperative schema); or durably stores user's Inbox and Outbox, which are referencing the messages on replicated nodes (advanced cooperative schema). One interesting aspect of the system is the hash function algorithm, which decides where to replicate the email. The algorithm takes into consideration on-line habits, workload, and trust relationship of peers to select optimal nodes for the email replication. The weakness of the system is in the use of the super nodes, which to some extent is a single point of failure, and requires data replication when rebuilding the failed super node on another super node. Also, this schema may not work well when used outside of the community environment since the super node failure and recovery will not scale.

In Decentralized Electronic Mail (DEM) Bercovici et al. [23] architecture, all primary components, like user's mailboxes, are mobile objects distributed and replicated over multiple participating computers. Dispatch unit, also a mobile replicated component, maintains a list of connected hosts and unique references to mobile objects. Attachments are objects too, allowing multiple emails to point to a single attachment. Mail items travel directly from senders to the receivers achieving *O(1)* communication. Replicas in DEM perform periodic synchronization. The issue with DEM is that the basic mobile object is the whole mailbox, except for the attachment maintained in the mailbox via the reference. Consequently, replicating the mailbox or moving it to the connected client or moving it from the disconnected client is expensive.

ePost in Mislove et al. [110], stores messages and metadata in a peer cooperative store maintained by DHT. The messages are posted to storage and the recipient is notified via Scribe multicast system, also DHT based. When the user is on-line, she receives the notification and retrieves the message. The messages are immutable. A single-writer log, stored by each ePost user, maintains a user's view of the data in the system; i.e., the mailboxes. Email data and immutable log elements are self-authenticating via a Merkle hash tree. ePost uses logs to reconcile inconsistencies resulted from the network partition. ePost is one of the few systems that maintains a history of changes in the log and periodically takes a snapshot to reduce the overhead of the log traversal. ePost garbage-collects deleted messages. ePost is the most comprehensive P2P architecture, which was deployed for two years in real user's environment, Mislove et al. [109]. While ePost might be a viable solution as an internal application within an organization, it is not obvious how well it works as the global email service. There are many issues that have to be resolved in this case. For instance, initial bootstraping, Public Key Infrastructure (PKI)[10], administration, maintenance, user's incentive, and defense against attacks may be problematic. There is also a question of a users' entrusting her data, albeit encrypted, to complete strangers.

Kageyama et al.[83] use DHT to forward notifications and control information from senders to receivers. Messages and public keys are stored on peers and could be replicated within a trusted group to improve availability. The architecture is a pull-based. Notifications with partial headers are sent to the receiver's key in DHT. The receiver periodically

[10]`https://en.wikipedia.org/wiki/Public_key_infrastructure`

polls DHT and either pulls the message or deletes it without downloading. This way the overhead of the Outbox storage has to be addressed by the sender, which makes the email SPAM less viable. While using pull rather then push, this architecture is not significantly different from ePost. To some extent ePost is pull-based as well. Indeed, in ePost the outgoing message is replicated between peers. Notification then is sent to alert the recipient about the message. The recipient may ignore the notification, thereby offloading the burden of initial email storage to the peers. The difference is that ePost picks random peers whereas Kageyama chooses trusted peers. But in ePost all notification messages are signed by the sender, which allows building efficient Spam block lists.

In the Distributed Mailing System (DMS), Mezo et al. [106], contribution of each peer to the email system is based on the evaluation of peer resources. The result of the evaluation classifies peers into entities and super nodes, increasing overall performance and reliability. DMS maintains a hierarchy of three layers: dispatch, community, and entities. Each layer maintains and replicates data according to its function, with the lowest entity layer replicating the email messages, community handling inter and intra community communication, and dispatch handling communication between communities in different regions and has a subset of super nodes registered in DNS for interconnecting communities across countries. DMS's classification of piers and community layering makes resource usage efficient but in addition to other issues present in P2P architecture, susceptible to targeted attacks, for instance Denial of Service (DoS) attack targeting the dispatch community.

HMail is a hybrid of traditional and P2P system, Mezo et al. [107]. HMail peers are layered into a hierarchy of three layers: 1) base Chord overlay; 2) peers with higher uptime and bandwidth availability validated by GeoIP tagging; 3) peers with higher processing power and storage selected from the nodes of the second layer. Layer three is partitioned into Spool, Inbox, and the activity monitor. The activity monitor maintains PGP keys. Partitions are used by nodes within the same geographical area. HMail separates the decision tasks that facilitate the mailing operations over several hierarchical blocks distributed geographically. Communication between components occurs in restrained, hierarchical manner. A traditional email system collaborates with P2P to accomplish email functions. Daemon application running on layer three provides SMTP/POP3 functionality. To provide interoperability some nodes from the spool are registered as MX-hosts in DNS. HMail implements higher level of hierarchical organization than DMS but in general suffers from the same issues as DMS. Since HMail supports a traditional centralized server, it inherits its weaknesses as well.

In summary, P2P approach uses peers' resources to implement store-and-forward architecture of the centralized service and deliver high quality of service. In order to accomplish these requirements, it must have both replication of the delivery path and the content. Depending on the peer's availability, this may require substantial resource allocation especially in the latter case. Resource-based hierarchy and geographical distribution of peers may improve the replication's performance. Key-based routing in DHT, which is characteristic of P2P architecture, puts additional stress on P2P resources as the content and the metadata are moved through the intermediate nodes. This could be alleviated by maintaining references to mobile Inbox and Outbox, directly connecting them when needed. Since a content is stored on untrusted peers, a robust security mechanism must be implemented, including the PKI support. Incentive mechanism has to be put in place in order to motivate users to contribute their resource in a fair way and provide truthful estimate of their capabilities. While P2P architecture has attractive properties (like organic growth, where as more peers join in the more resources are added to the network, and common content namespace so that attachments can be shared amongst peers), it requires a complex adaptive infrastructure. There are successful P2P applications like BitTorrent or Bitcoin[11], but to-date P2P found no traction in email outside of academic interest. Perhaps, moving the user's data into potentially "untrusted" peer's devices, where while encrypted it could be accessible by the world, is not any more attractive to consumers than having the data in central location where it is data-mined by the "trusted" email provider.

2.1.3.3 More on centralization

The related work list would not be complete without Apache Wave (originally Google Wave) [7]. Google Wave by Google is an ambitious architecture merging instant messaging, email, Wikis, and social networking under the web-based computing platform. Google Wave messages (waves) with their complete threads (blips) are perpetually stored on a central server, making it another variation of the "Siren Server". Waves support concurrent modification and low-latency updates and are shared with collaborators who can be added or removed at any point in wave's existence. The history of each wave is stored within it. Google Wave provides federation over Extensible Messaging and Presence Protocol (XMPP) extension. Google Wave somewhat resembles ideas presented in Bayou, in particular Timeline. Surprisingly, Google Wave was not successful, with the failure attributed to an overly complicated interface resulted from the unified platform yet without any apparent benefit over existing solutions, Stokes [138].

It is interesting to note that while P2P research emphasizes centralized architecture disadvantages and a way to solve them with P2P architecture, there are research papers that use centralized email servers because of their high availability, backup, redundancy, and free space. Mr.Privacy builds social networks on top of email, Fischer et al. [60]. Mailbook is another social application that uses email to build a P2P network, Yong et al. [156]. Choi et al. [40] uses email as the way to submit a request to an offline peer. MailZoro is an email based P2P file sharing protocol, Dhiwal et al. [48]. Srinivasan et al. [137] aggregates back-end storage by establishing a RAID-like system on top of virtual email disks formed by email accounts. Clearly, email providers cannot contribute their resources for free and there must be a tipping point at which these kind of applications are either blocked or providers change their business model and start charging a fee. I see a business opportunity for email providers to use the strength of their infrastructure, where if an alternative decentralized email architecture gains popularity, they can sell the email domain, DNS, and PKI services to users. For instance, if Alice has an account with *Acme* provider, her email address can still be *alice@acme.com* but rather than storing Alice's email, *Acme* resolves Alice's account name to a list of Alice's owned devices that actually handle her email. Likewise, it distributes Alice's public key for the email encryption.

2.1.3.4 Distributed file systems and delay tolerant networks

Regardless of what the email's architecture is, it can be broadly considered as a distributed asynchronous collaborative message sharing application. The application has append only storage where email messages are immutable once created. Updates to the email's

[11] `https://bitcoin.org/en/`

metadata, for instance mailbox name, are maintained by revisions and snapshots, and the data are eventually synchronized between clients and servers. In addition, email has to be replicated to provide availability and reliability. Generally, these basic properties are characteristic of a distributed system, and can be provided by DFS. Therefore, it is beneficial to review the related work on DFS.

Satyanarayanan in [130] reviews common properties and classification of DFS. He classifies computing models into four levels from the perspective of the FS design. The first level is single-user at a single site running one process. The FS for this model addresses four key issues: the naming structure of the FS, the application programming interface, the mapping of the FS to the physical media storage, and the integrity of FS across various failures. The second level is a single user running multiple processes on a single site. This adds concurrency control to the FS design considerations. The third level, a classical time sharing system, is multiple users running multiple processes on single site. Security becomes an important design consideration. The final level, the DFS, has multiple users dispersed in a network of autonomous computers sharing the same FS. The challenge is in realizing this abstraction in an efficient, secure and robust manner. In addition, the issues of file location and availability assume significance. An approach to file location is an explicit mechanism mapping file names to storage sites. Availability is of high significance because the usage site can be different from the storage site. Replication is the mechanism to achieve availability but introduces a problem of its own since changes have to be propagated to all replicas in a consistent and efficient manner. Over the years the DFS has evolved with features like network transparency, support for atomic transactions on files, and file caching. A number of empirical observations has been made about DFS:

- Most files are small, under 10 KB.
- Read operations are much more frequent than write.
- A random file access is rare.
- Data in files tend to be overwritten often.
- Most files are read and written by one user.
- If a file is referenced, there is a high probability it will be referenced again in the near future.

Some of mechanisms to be found of values in DFS are:

- Mount points to provide applications with a single, seamless, hierarchically structured, name space.
- Caching of data at clients that contribute most to the performance in DFS. Cache can be in-memory or on a local drive and either pages or an entire file can be cached. Cache validation can be done by the client polling the server or the server sending notification just before the cache is invalidated. The latter is more efficient in terms of the bandwidth.
- Hints can substantially improve caching performance if correct and have no negative consequence otherwise. Hints are mostly used for file location information.

- Transferring data in bulk reduces the overhead caused by protocol processing.
- Encryption is of primary value in preventing unauthorized release and data modification.

Some of the issues that are active subject of research in DFS:

- Availability as it is expected for DFS to be resilient to failures. Replications and disconnected operations are some of the mechanisms to achieve this goal.
- Scalability causes a number of issues like system management, network congestion, use of single hierarchically-organized name space, and complex network topologies.
- Heterogeneity is inherently difficult because of the multiple computational environments, each with its own notion of file naming and functionality.
- Database access is difficult because the applications that use distributed databases usually demand strict consistency of data as well as atomicity of groups of operations. Distributing a database is particularly difficult at large scale. A simplified approach is to provide a distributed access to a data on a single server.

Ghemawat et al. in [64] present a scalable distributed Google File System(GFS) for large distributed data-intensive applications. Key design considerations of GFS are:

- Component failure is the norm rather than the exception. GFS consists of thousands of inexpensive commodity parts. Constant monitoring, error detection, fault tolerance, and automatic recovery are integral to the system.
- Files are huge. Multi-GB files are common.
- Most files are mutated by appending new data. Appending is the focus of performance consideration.
- Co-designing the applications and the FS API benefits the overall system by increasing flexibility. Consistency model is relaxed. Atomic append operation is introduced, eliminating need for synchronization.

Files are organized hierarchically and identified by path name. GFS supports snapshot and record append operations. The snapshot makes a copy of a file or a directory tree and is used to create branch copies or checkpoints. Files are divided into fixed size chunks (64 MB) identifiable by an immutable unique chunk handle. For reliability, each chunk is replicated between replica servers, which is three by default. There is one single master server maintaining the metadata like the file and chunk namespaces, mappings from files to chunks, and current chunks location. All metadata is stored in master's memory. Namespaces and mappings are also stored on the local drive and are replicated to remote servers. Master keeps the metadata up to date via heartbeat messages to chunk servers. Neither the client nor the chunk servers caches data because most applications stream through files too large for caching. The operations log maintains a historical record of critical metadata changes. Files and chunks, as well as their versions, are identified by logical times at which they were created. The log is replicated for reliability. The master checkpoints the log for faster reloading. GFS applications handle the relaxed consistency by relying on append rather than overwrite, checkpointing, and writing self-validating, and self-identifying records. Modification to chunks are handled by an expiring lease granted by the master server to a process with no other processes having access to the chunk. The changes are propagated with the backup copies by the primary chunk holder to other chunk servers. Changes are saved after all chunkservers acknowledge, guaranteeing atomicity.

Braam in [31] presents the Coda DFS. The main focus of Coda is on the highest degree of availability in the face of all realistic failures. This is accomplished by mechanisms of disconnected operations and replication. In Coda, when a file is accessed by a client, the file is cached on the client's local drive. All consequent operations on the file are performed on the local copy. Directories are also cashed on the client. Modifications to the files and directories are propagated to the Coda servers. Empirical studies show that the files modifications are rare as compared to read, consequently Coda heavily relies on caching to improve the performance. When updates have to take place in the connected mode, the files are propagated to the server. In disconnected mode the updates are stored in the client modification log (CML). Upon re-connection, CML is integrated on the server, and the replay of the updates from CML brings the server up to date. If there is a conflict, e.g. another client made modifications to the same file, then a repair or merge is needed. Sometimes the repair can be automatic, for instance if one client inserted an appointment into the calendar for Monday and another client inserted the appointment for Tuesday. In other cases the user's intervention is needed to repair the conflict. Coda has replication servers and updates are generally made to all servers. Replication provides high availability of data. A disconnected server is eventually updated with the changes. Coda does this by requesting a time stamp from all servers when a file is requested. If the server does not have the latest copy of the file then a resolution process is initiated which tries automatically to resolve the differences.

Gazagnaire et al. [63] introduces Irminsule. Irminsule is a branch-consistent library database and is based on Git[12], a Distributed Version Control System (DVCS). It is designed to solve issues raised by the CAP theorem. To do this, Irminsule provides a collection of libraries for database primitives. The libraries can be used to implement a distributed application running on heterogeneous devices. Depending on the device type, various policies can be implemented, for instance:

- Choose to store a subset of data in the Cloud, with the backup on the user device, like settop box.
- Define the level of trust; for instance, data in the Cloud should be encrypted.
- Define synchronization schedule. For instance, some devices can be synchronized only when WiFi is available.
- Provide history of changes for monitoring and debugging.

Irminsule provides high availability by getting rid of strong consistency. Each device may have its own replica, either full or partial. The latter corresponds to a branch in the global database. Reads and writes are local. Merges with the global database or other devices happen at a time controlled by the application. Since Irminsule is based on Git, the data in the store are immutable. As consequence, the store is expected to grow but it can be managed with data compression and garbage collection, and it might not be a problem

[12]`https://en.wikipedia.org/wiki/Git_(software)`

since the commodity storage is getting cheaper. Irminsule can be used to implement the FS[13].

In summary, DFS provides valuable mechanisms to support decentralized email architecture. Since email is a distributed application, availability is of high significance. Both GFS and Coda have as their design goal, handling of frequent component failure and network interrupts. Both FS's tune their performance for read, and append in GFS, rather then frequent update, which is consistent with the email's model of immutable storage. Coda appears to be more email "friendly" by providing both application and server level data repair and resolution after the network reconnect, replicating full files rather than file chunks, and by not having master server maintain the namespaces and metadata. But GFS is designed for the datacenter distributed environment, whereas Coda incorporates many features suitable for mobile computing. While GFS, or Coda, or generally speaking any capable DFS, can support decentralized email architecture, they also present additional overhead and level of complexity. On the other hand, a collection of libraries, like Irminsule, specifically designed to run on heterogeneous devices and support various flavors of data consistency presented by CAP, might be a less complex and more efficient solution from the user and application designer prospective.

Email is an asynchronous messaging service with the sender and receiver not required to be online at the same time. Consequently users of email are tolerant to delays. Delay is inherent to the email's store and forward architecture. In this respect, the DTN is similar to email. "In a sense, the DTN architecture provides a common method for interconnecting heterogeneous gateways or proxies that employ store-and-forward message routing to overcome communication disruptions. It provides services similar to electronic mail, but with enhanced naming, routing, and security capabilities" - RFC 4838 [37, p4]. This similarity makes DTN a good fit for email implementation in cases where either the client or the server have intermittent connection to the network or are disconnected for long period of time. Husni and Wibowo in [79] propose an DTN based architecture to provide email to remote villages that do not have an Internet connection. The villages have train service with a central station having the Internet connection and all stations having a WiFi connection. When the train stops at the station without an Internet connection, it downloads to the train router the messages from the village addressed to recipients outside of the village and uploads to the train router the messages addressed to the recipients at the village. When the train stops at the central station, it downloads to the train router the messages addressed to the recipients from the villages along the train's route, and uploads to the station router the messages addressed to the recipients at the global Internet. In addition to uploading-downloading messages and providing the temporary storage for the messages, the train router converts the messages to-from bundles. The bundle is the series of contiguous data blocks routed in DTN via the bundle layer [37, p3].

Bin Sa'Adi in [26], offers a DTN email architecture to provide the email to remote villages similar to [79]. The architecture consists of the online email server connected to the Internet and the offline email server in the village. Android smartphone, owned by a villager, works as the data mule, loading and unloading the email messages when it arrives at the terminal location. Synchronization of data between the smartphone and either server is handled by FreeFilesync[14]. In both architectures [79, 26], the train's router

[13]`https://github.com/mirage/irmin`
[14]`https://www.freefilesync.org`

and the vilager's smartphone provide temporary storage or caching for the email message as part of the store-and-forward mechanism. In the centralized email architecture this functionality is provided by the SMTP server. I assume there is a direct connection between the sender and the recipient in the decentralized email architecture based on IoT environment, which is different from the DTN email architecture. Nevertheless, the caching mechanism can be used in the decentralized email architecture in the case when, for instance, the email message is received by the recipient's smartphone. Then the message can be cached until it is opportunistically backed up to the IoT or the virtual cloud device. Details of the high-level decentralized email architecture are given in Section 2.2.

2.1.4 Problem formulation

The question that I raise in this chapter is whether we can take advantage of the emerging IoT technology to address some shortcomings in present day email. There are many aspects to this problem and it is not possible to address all of them in this thesis. What I am attempting to answer is whether it is feasible to have the decentralized email system with certain properties running on the resource-constrained device, which is characteristic of the IoT environment. As I suggested in the introduction to this chapter, maintaining email revisions addresses some of the issues inherent in a distributed system like email. Moreover, revision control can be part of any distributed system implementation where divergent replicas have to be eventually merged and reconciled. This feature is especially relevant to collaborative systems, for instance Slack[15] where the history of a document change is important. I am, therefore, proposing a high-level decentralized email architecture in Section 2.2 with emphasis on revision control. I then evaluate this architecture in Section 2.4 for the basic email functions of appending, fetching, and synchronizing message in the email archives.

2.2 High-level email architecture

The architecture is based on the previous research where email history persistence is one of the features, most notably Google Wave and to some extent Bayou and ePost, and the research into email decentralization via P2P network. In a P2P decentralized email architecture, a user contributes the resources of her device to achieve availability and privacy but the user's data is distributed over a number of generally untrusted devices, the management of the system is complex, and it could be susceptible to a number of attacks. The novelty of this research is owed to IoT phenomena "in which a continuum of devices and objects are interconnected with a variety of communication solutions such as Bluetooth, WiFi, ZigBee, and GSM, to name a few. Telefonica estimated that 90 percent of cars will be connected to the Internet by 2020" [154, p2]. Cisco predicts 27 billion of connected devices by 2021. Consequently, in IoT environment a user is the owner of multiple trusted devices which provide availability, backup, and intrinsically privacy. The email architecture uses and extends, when needed, the IoT infrastructure of communication, storage, management, and security.

IoT and other devices can be shared by family members or trusted friends, increasing the disk usability, and reducing the network bandwidth by transmitting a duplicate at-

[15] `https://slack.com`

tachment only once. Attachments take up a significant share of the user's email archive. I have analyzed attachment's statistics of the email datasets used in the research (Section 1.1). Enron's attachments in average contribute 91.21% to each user's archive with standard deviation (SD) 9.18% and coefficient of variation (CV) 0.1. Duplicate attachments in average contribute 14.9% to each's user's archive with *SD* 12.03% and *CV* 0.81. Overall size of unique messages in Enron corpus is 23.83 GB with duplicate attachments taking up 34.8% of the space. The overall percentage of duplicate attachments is higher because there are attachments shared between users. In private email archives, the numbers are on the same order of magnitude with the average attachment contribution 81.93%, *SD* 18.65%, *CV* 0.23; duplicate attachment contribution 13.84%, *SD* 8.78%, *CV* 0.63; overall unique messages size 68.2 GB with duplicate attachments taking up 27.22% of the space (Table 3.1). It is therefore beneficial from the disk space, energy, and bandwidth prospective to store and transmit messages as MIME parts, where the attachment is stored as an individual file. There could be additional energy savings if attachments are saved and transmitted as the binary instead of base64 encoded, with former producing 33% smaller data image at a lesser energy cost than compression. Socially-connected users can increase the savings by sharing their globally accessible devices. I evaluate the cost saving from device sharing in Section 3.7.

Figure 2.3 shows the high-level architecture example. The architecture makes the following core assumptions:

- User owned devices are categorized into two groups. The first group is mobile; for instance, smartphone, tablet, laptop, smartwatch, or connected car. The second group is stationary; for instance, desktop, home router, or home energy monitor.
- A user owns one mobile device and a stationary device. According to Pew Research Center 90% of U.S. households owned in 2017 at least one of smartphone, desktop/laptop, tablet, or streaming media device, with the typical (median) household containing five of them[16].
- A user has a home broadband service and owns or rents the network router. According to Pew Research Center 73% of U.S. adults had broadband service at home in 2016[17].
- Stationary devices (network router, desktop, etc.) are always on or wake up due to network activity. The former is true for a network router or an energy monitor and the latter applies to a desktop/laptop.
- All devices have storage capacity that can be expanded as needed to support email backups. The storage cost is affordable at only a few cents per GB for the hard drive[18]. Cloud storage is also affordable with some cloud providers offering free storage up to 20 GB and monthly plans priced at a fraction of a cent per GB[19]. Prices on MicroSD card storage for smartphone and tablets are also affordable at $44-128 for 128 GB and 256 GB cards.

[16]http://www.pewresearch.org/fact-tank/2017/05/25/a-third-of-americans-live-in-a-household-with-three-or-more-smartphones/

[17]http://www.pewinternet.org/fact-sheet/internet-broadband/

[18]https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte/

[19]http://www.zdnet.com/article/cloud-storage-price-check/

- Communication between stationary devices is handled via the home network, either Ethernet of WiFi.
- Communication between mobile and the stationary devices with a globally accessible address is handled either through the Cell or opportunistically via a WiFi network when away from home and via WiFi when at home.
- Mobile and at least one of the stationary devices can be directly connected to another user's mobile and the stationary devices. This can be, for instance, accomplished via the Signpost [125]. Another option is for the devices to support globally addressable IPv6 unicast address. According to the network operator measurement as of December 13, 2017 leading USA mobile and cable providers T-Mobile USA, Verizon Wireless, ATT, T-Mobile USA, Sprint Wireless, and Comcast IPv6 deployment was 87.88.21%, 80.88%, 67.34%, 62.65%, and 60.98% respectively [20]. It is expected that the IPv6 user count will exceed 50% by 2019[21].
- User's email is resolved to her globally accessible devices prioritized in order of preferred connection. This can be handled, for instance, with Signpost [125].
- When sending a message to the email list, the message is opportunistically sent via WiFi to reduce the energy cost but the delivery could be relayed to the globally accessible home IoT device, which in turn distributes the message to the email list.
- The email message is end-to-end encrypted with a session key, which in turn is encrypted with the user's public key. More details on the message encryption is provided at the end of this section.
- Conceptually, email clients function at the application layer the same way as in the centralized architecture and do not require additional storage or connection. But mobile devices, for instance a smartphone or tablet, could be configured by the user to provide email backup if the devices have enough storage capacity.
- The user may choose to have cloud storage for email backup.

Let us consider two groups of devices in Figure 2.3. The first group consists of Paul's devices and the second group consists of Alice's and Bob's devices. Each group has multiple globally accessible devices. Paul has a Network Router, Smartphone, and WiFi car with the globally accessible address. In addition he has a laptop and tablet. Alice and Bob have a Network Router, Smartphones, and Cloud backup with the globally accessible address. In addition, they have a desktop and Bob owns a WiFi Smartwarch. Each device has storage which either maintains a full email replica with a history of changes or functions like a common email client (marked with (c) next to the storage). Paul's router, laptop, tablet, and his car and Alice and Bob's router, desktop and cloud storage provide full email replica. All of their smartphones and Bob's smartwatch function like a common client. Alice and Bob are socially connected to Paul and actively exchange emails with each other. Consequently, some of their devices are configured to maintain replica of both of their email archives providing availability, redundancy, backup, and efficient disk usage. For instance, Paul can access his email via his smartphone from either his or

[20]`http://www.worldipv6launch.org/measurements/`

[21]`https://www.internetsociety.org/resources/doc/2017/state-of-ipv6-deployment-2017/`

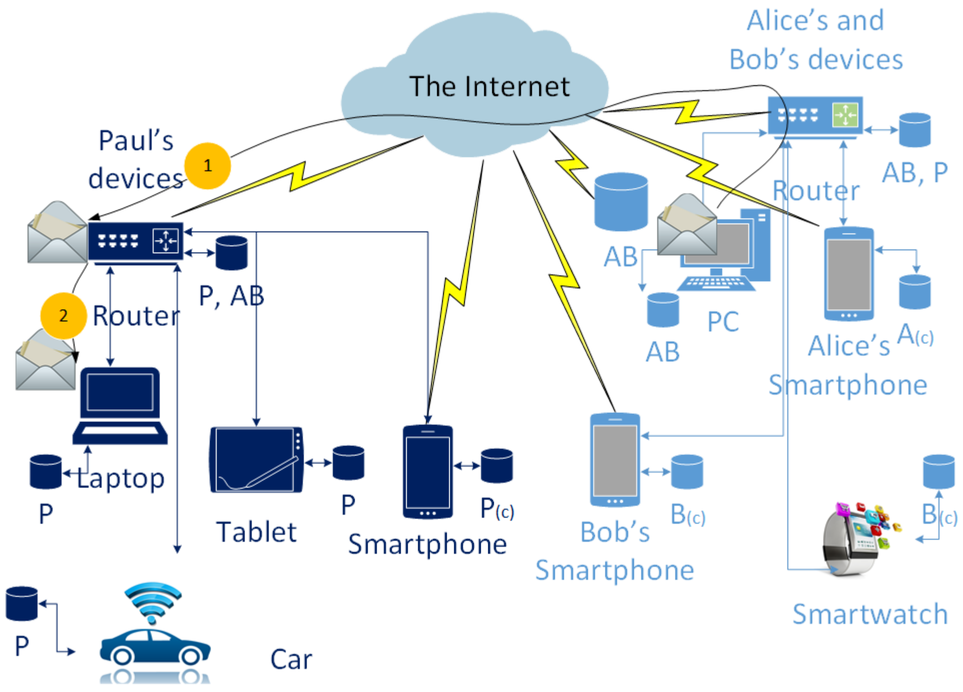


Figure 2.3: High-level decentralized email architecture.

Alice and Bob's router. When at home, Paul can access his email by directly connecting to his router via WiFi.

Consider the following email application use cases:

- When Bob is at home and wants to send an email to Paul, Bob composes the message on his desktop and sends the email to paul@acme.net. paul@acme.net is resolved to Paul's router, and email is sent directly to Paul's router (1). Paul happened to be at home and reads his email, downloaded from his router, on his laptop (2). The email is synchronized via WiFi to all devices maintaining full email replica.

- Paul is in the office and wants to send an email to Alice, as shown in Figure 2.4. He composes the email on his laptop and sends the email to alice@wonderland.io. Alice's email address is resolved to Alice and Bob's router and the email is sent directly to the router (1) where it is replicated via WiFi to all devices maintaining full email replica. The email is also replicated to Paul's devices (2). Alice accesses the email via the office's router (3). If the link to Alice and Bob's router is down, then the email is sent directly via the Cell network to Alice's smartphone (1a). When Alice comes home, email is synchronized with other devices maintaining full email replica (1b). If the link to Paul's router is down but Paul has his tablet with him then the email is synchronized with the tablet (2a) and then later with all other devices maintaining full email replica when Paul comes home (2b).

- Finally, Bob is at the airport and wants to send email to Paul, as shown in Figure 2.5. Bob only has his WiFi-enabled smartwatch on him. Bob speaks his email into the watch and asks the watch's personal assistant to send it to Paul. The link to Paul's router is down. Paul's smartphone broke and Paul is driving to the mobile store to buy a replacement smartphone. But first Paul has to fill his car with gas

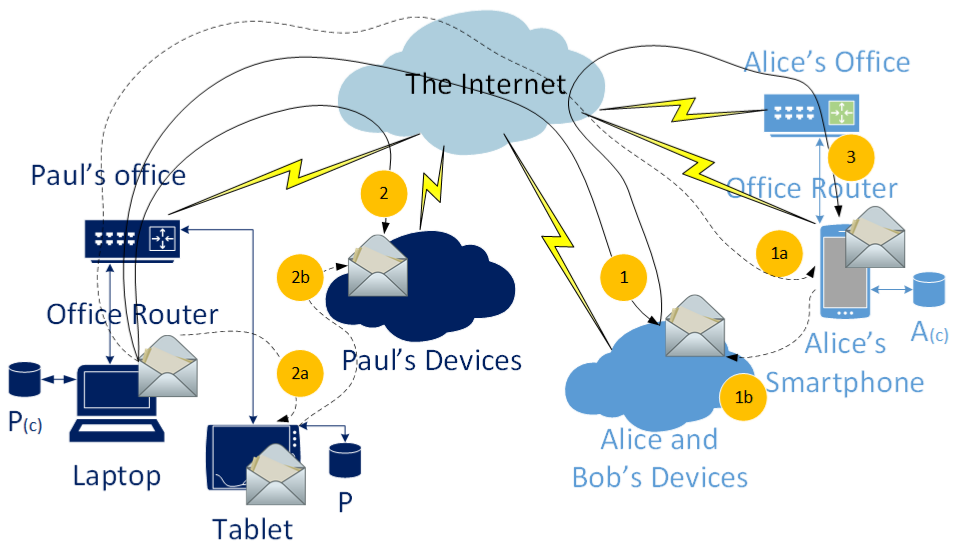


Figure 2.4: Sending email from the office.

and he stops by a gas-station. In the meantime, Bob's watch tries to connect to Paul's globally accessible devices. While at the station, Paul's car connects to the gas-station's WiFi router and Bob's watch succeeds in connecting to Paul's car and uploading the email message (1). Paul listens to the email read by the car's personal assistant. In the meantime, Bob's watch also uploads the email to Alice and Bob's cloud storage (2) because the link to their router is down. Eventually, when the link is up, the email is synchronized with other devices maintaining full email replica (3). The link to Paul's router is also down and the email is synchronized with other devices via WiFi when Paul is back home and parks his car in the garage (4). If the link to Alice and Bob's cloud storage is down, then the email can be temporarily uploaded to Alice's phone (2a) and, when Alice comes home, synchronized with other devices via WiFi (3a).

In all cases, the architecture addresses disk and bandwidth optimization, data confidentiality, and eventual consistency in the following way:

- Sending or synchronizing email does not transmit attachments already existing on the remote device saving energy and bandwidth.
- MIME parts are transmitted in a prioritized scheme. MIME metadata and headers are transmitted first and attachments are transmitted as required. Email is stored as content-addressed MIME parts. This provides single storage for duplicate attachments, saving energy and disk space.
- Attachments are transmitted and stored as binaries rather than base64-encoded. This eliminates the need for compression and saves energy and bandwidth. All other MIME parts are compressed.
- To preserve a user's privacy, attachments are encrypted with convergent encryption, Douceur et al. [53], which maintains single storage across multiple archives. Other MIME parts are encrypted with a random session key. The session key is encrypted with the user's public key. Details of encryption and PKI are outside of the scope of this research's.

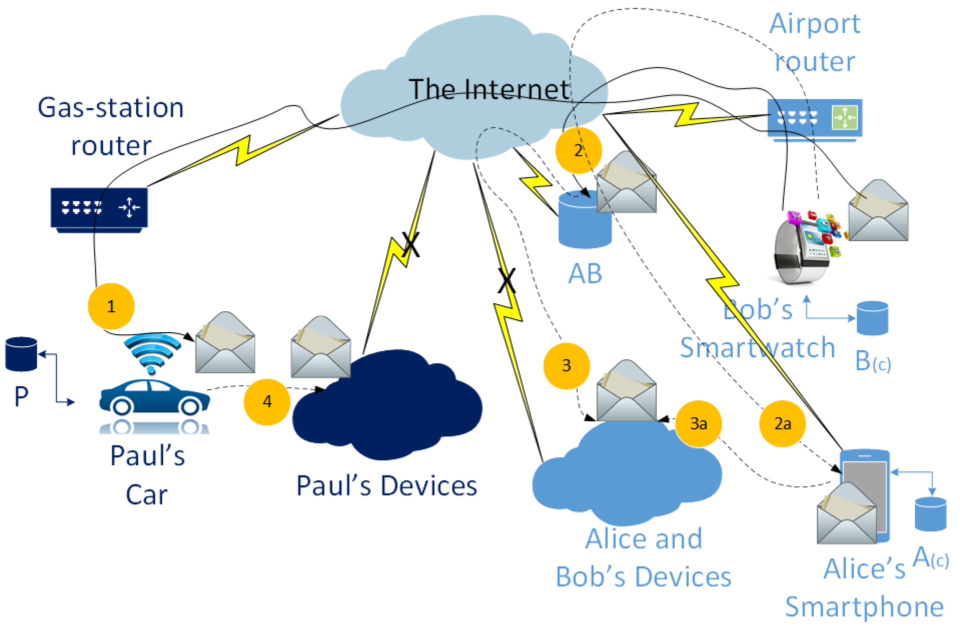


Figure 2.5: Sending email on the move.

- Each email replica maintains the complete history of changes. This enables the replicas to converge (merge) at some point.

The remaining sections in this chapter discuss evaluation's methodology (2.3), analysis of the data (2.4), limitations of the methodology and evaluation (2.5), availability (2.6), and architecture's energy cost evaluation (2.7), .

2.3 Evaluation methodology

2.3.1 IoT device used in the evaluation

The architecture described above is composed of traditional devices like smartphone, tablet, laptop, etc. and physical objects of "things" comprising the IoT. The "things" represent a wide spectrum of devices constrained by the processing power, memory, and power consumption. These devices, in addition to collecting data, also manage and consume the data, for instance the Databox [73]. One of the devices, representative of the resource-constrained device characteristic of the IoT environment is Raspberry Pi. Raspberry Pi is a credit-card sized single board computer developed in the UK with the intent of promoting the teaching of basic computer science[22]. As of February 2016 there were eight million devices sold making it the best-selling UK personal computer. Raspberry Pi supports various Linux OS flavors and Windows 10 IoT core. At a cost of $35 and a capable hardware and software environment, Raspberry Pi has inspired a community of enthusiasts engaged in different type of projects, including IoT[23]. While there are other

[22] `https://en.wikipedia.org/wiki/Raspberry_Pi`

[23] `http://www.informationweek.com/software/enterprise-applications/10-raspberry-pi-projects-for-learning-iot/d/d-id/1320757`, `https://developer.microsoft.com/en-us/windows/iot`, `http://www.ibm.com/internet-of-things/ecosystem/devices/raspberry-pi/`, `https://www.raspberrypi.org/blog/tag/internet-of-things/`

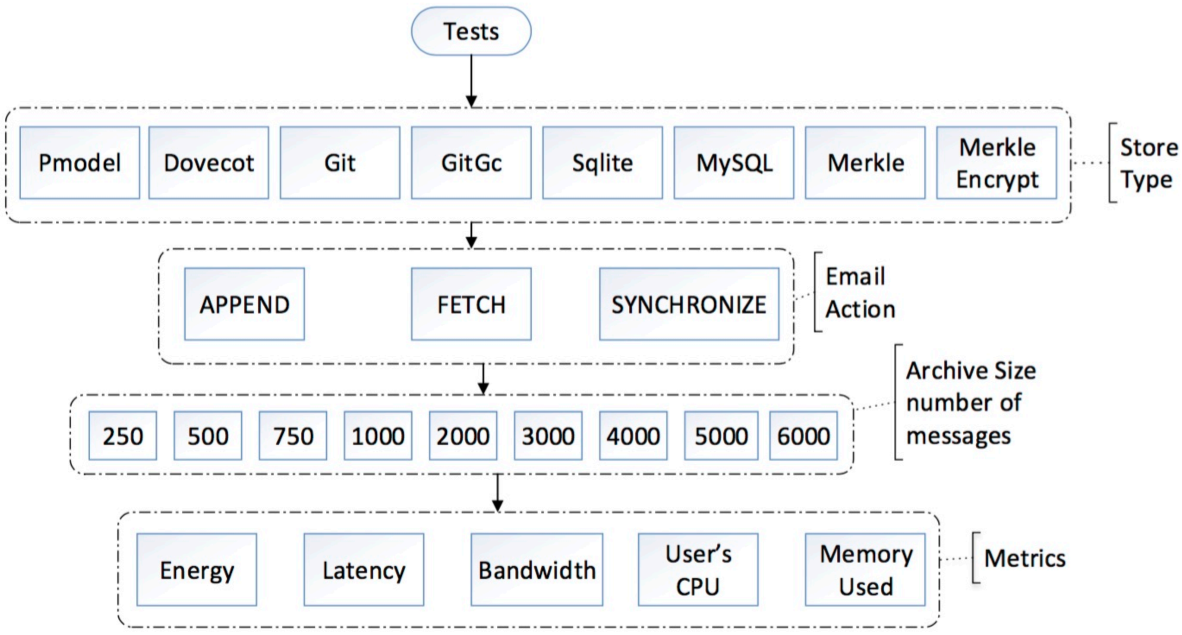


Figure 2.6: Evaluation workflow.

competing platforms that emerged after the success of Raspberry Pi[24], Raspberry Pi has the best support community. Consequently, I used a Raspberry Pi in my evaluation. The latest version of Raspberry Pi 3 has a quad-core 1.2 GHz processor, 1 GB of RAM, 10/100 Ethernet or 802.11 WiFi, and MicroSD card storage. The Raspberry Pi used in the evaluation has the official Raspbian OS installed on 128 GB MicroSD card.

Hameed et al.[74] evaluate Raspberry Pi as an affordable, lightweight, and energy-efficient private email infrastructure. The authors evaluate Postfix SMTP and Dovecot IMAPv4 servers running on Raspberry Pi 2 under various email loads. Authors conclude that the Raspberry Pi is an adequate platform for individual or small and medium enterprises with up to 4 000 email load per day. This evaluation validates the idea of employing a user's resource-constrained devices like her home router, power monitor, tablet, or smartphone as a hardware platform in the email architecture supporting revisions, data replication and efficient synchronization, while intrinsically providing a user's privacy.

2.3.2 Evaluated email functions

Regardless of the email application use, the data structure, or the encryption, email conceptually must support three functions - appending new messages to the user's mailbox, reading the messages, and synchronizing the messages between user's devices. Consequently, these functions are the focus of this analysis.

2.3.3 Email back-end used in the evaluation

Maintaining email revisions enables replicas to converge at some point in time per CAP theorem [59, 34, 63]. Consequently, I evaluate back-ends which implement revisions and

[24]http://www.computerworlduk.com/galleries/it-vendors/move-over-raspberry-pi-9-single-board-computers-for-geeks-3544497/#10

compare them with systems that do not. I run Dovecot server version 2.2.24[25], as the base email system in the evaluation. Dovecot is the open source, high performing IMAP server supporting various Linux platforms and MAC OS X. To model different store types, I built a TCP/IP server that implements a subset of IMAP commands - APPEND and FETCH. I have chosen to write the server in C++ because there is a wide support for third party API's, OOP concepts allowing for easy plugin of various store types, comparable performance platform with Dovecot, written in C. I implemented a number of back-end stores, with and without email version support. I used level 6 compression on all store types and whenever possible *fdatasync*[26], to force all modified in-core data to be written to disk, so that the energy and the latency measurements are more predictable. Without *fdatasync*, the OS may commit data to the disk after application program completion, making it appear to run faster and have less energy consumption. For simplicity and to make evaluation of different store types comparable with the base Dovecot system, which does not support on-disk encryption (though it is possible to implement it with Sieve filter), I do not encrypt messages on disk. Although I do run one test to show encryption overhead. I also do not parse messages into MIME parts.

Overall I evaluate eight email back-end store types:

- A Maildir-like store (Pmodel) in which each message is saved to a single file. This store is used as the validation point for other store types and as a lower boundary estimate since it provides a simple conceptual email implementation. The Pmodel does not implement versioning. I use *rsync*[27] to synchronize local and remote Pmodel stores.

- Dovecot is the base system with the Maildir store. Dovecot does not implement versioning. I use Dovecot's *dsync* daemon to synchronize local and remote email archives.

- Git store. I chose Git because of its emphasis on performance, non-linear workflow, efficient synchronization, and revision maintenance via the Merkle tree with content-addressed storage as content's SHA1. There are other DVCS that use SHA1 hashes to maintain revisions but evaluating all of these systems is outside of this research's scope. Git has properties common to any Version Control System (VCS) like branching, merging, re-visioning, and is particularly good at lightweight snapshots. I envision a Git-like system or VCS as an ideal solution for an email back-end and versioning. My implementation of the Git store, from the logical message structure point of view, is conceptually similar to the Maildir. Messages are stored in individual files with the metadata encoded in the file name. I take advantage of the Git ordered tree object and use it as the index to the messages. To make an efficient implementation of the Git store, I made a simple change in Git source code to receive commands over TCP/IP instead of the command line and I pass commands directly to the built-in functions. I use Git's *fetch* command to synchronize local and remote Git repositories.

- GitGc store is identical to the Git-one with addition of Garbage Collection (GC), which runs every 250 appended messages. GC compresses and packs loose object

[25]`http://www.dovecot.org`
[26]`http://linux.die.net/man/2/fdatasync`
[27]`http://linux.die.net/man/1/rsync`

Table 2.1: Evaluation Test: number of messages and respective archive size.

Number of messages	250	500	750	1 000	2 000	3 000	4 000	5 000	6 000
Archive size(MB)	41.18	54.18	64.85	122.65	256.4	442.8	574.27	604.05	831.74

into one single file. A corresponding index file contains the offset of each object in the pack file. GC is very efficient at delta-compression of revisions with the resulting pack file having little disk space overhead as opposed to loose objects.

- Sqlite[28] is a compact software library database. Sqlite is the most widely deployed database and is supported by smartphone platforms like iOS and Android and is suitable for embedded applications. Sqlite is also used as the back-end in Fossil[29], another DVCS similar to Git. Implementation of the Sqlite store is conceptually similar to the Maildir. The messages are stored as blobs. I take advantage of the standard Sql structured storage and relational integrity and keep the message's metadata and messages index in separate tables. Revisions are maintained via audit tables. I use *rsync* to synchronize local and remote stores since Sqlite does not have its own synchronization capability. Sqlite's database is a single file and *rsync* may have different performance as compared to Pmodel synchronization, which has multiple files.

- MySql[30] is an open-source relational database. In July 2013 it was the world's second after Sqlite the most used database[31]. MySql is an attractive back-end platform, which supports features commonly found in traditional relational database engines like SQL, transactions, ACID[32] compliance, replication, and clustering to name a few. Its performance therefore might be representative of other relational database performance on a resource-constrained computer. Implementation of Mysql store is identical to the Sqlite one. Mysql binary logs are used to synchronize local and remote databases.

- Merkle store is a naive VCS implementation. Messages are content-addressed via SHA1 and revisions are maintained in a single log file as Merkle's tree blockchain. This store has a lower revision's disk space overhead but may have a higher fetch latency since the index has to be inferred from the log file. I do not test synchronization with the Merkle store. I assume synchronization in this case is comparable to Git.

- Merkle-encrypt adds encryption to the Merkle store to evaluate encryption's overhead

2.3.4 Test design, data, hardware, and evaluation parameters

Figure 2.6 shows the evaluation's work-flow. Each test is repeated five times for an archive with number of messages and archive size as shown in Table 2.1. Each archive is randomly

[28]https://www.sqlite.org
[29]http://www.fossil-scm.org/index.html/doc/trunk/www/index.wiki
[30]http://www.mysql.com
[31]https://en.wikipedia.org/wiki/MySQL#cite_note-9
[32]https://en.wikipedia.org/wiki/ACID

generated from the Enron corpus email dataset. I use MAC OS X MacBook Air 1.7 GHz, 8 GB RAM, and 512 GB SSD as the client to append and fetch email messages and as the remote repository for synchronization with the local Raspberry Pi repository. Synchronization is tested by first appending 100 unique messages to already created archives and then synchronizing the archives to the original archives. The Raspberry Pi is connected to MAC via Netgear 10.100M FS608 switch. For each test I collect energy, latency, bandwidth, user's CPU, and used memory. Energy and latency are tracked with Monsoon Power Monitor FTA22D[33].

The Monitor has 4.54 V output, while a Raspberry Pi requires 5.1 V power supply. I considered it as an additional test to see how Raspberry Pi performs in under-voltage conditions, which may in fact reflect real life usage where USB power supply does not necessarily provide nominal voltage. By and large, the Raspberry Pi performed well, albeit ran slower, as it is expected with under-voltage. On a few occasions (generally under high CPU load) the Raspberry Pi switched to power-save CPU mode, which substantially increased the latency and the energy use of the test. This condition was clearly visible on the Power monitor and I re-tested those cases. CPU and memory are tracked with the *top*[34] Linux utility. Because some of the stores, for instance MySql, run as multiple processes, I pick the overall maximum user's CPU. To get the memory I calculate the difference between the lowest and the highest memory used during the execution, where the memory used is calculated as $(mem - free - buffers - cached)$. Bandwidth is calculated from *tcpdump*[35] output by adding up the payload packet's length. Error bars on each plot show 95% confidence intervals, which are illegible in some cases because of low variance in measurements. Large dots on all plots represent average values and the dashed line represents trend lines except for the memory and the CPU plots. The dashed line for these metrics connects the average values for better visualization of the results.

2.4 Evaluation

2.4.1 Appending Messages

I first analyze results for appending a set of messages via IMAP APPEND command. I use IMAP's APPEND command instead of SMTP server to add messages to the store so that there is no queuing by SMTP, which could make the latency and the energy measurement skewed. Figure 2.7 shows the energy used depending on the total appended archive size. As expected, the Pmodel performs the best, followed by Merkle, Mysql, Dovecot, and Sqlite. I also show, in Figure 2.8, the overhead from three computational tasks shared by all stores: receive messages from the network (net), compress messages (net+compr), and write messages to the hard drive (net+disk). As can be seen, all tasks have a linear trend-line with R^2 higher than 0.99, with compression task roughly contributing 2/3 and network-io with disk-io roughly contributing 1/4 each of the overall (net+compr+disk) Pmodel energy usage. We can also see that encryption (net+encr) introduces overhead of roughly 1/3 of the network-io energy use. One interesting observation is that if we add up energy contribution from each task ran separately - net-io, compression, and disk-io, then it appears about 10% higher than the overall energy usage when all tasks run

[33]`https://www.msoon.com/LabEquipment/PowerMonitor/`
[34]`http://man7.org/linux/man-pages/man1/top.1.html`
[35]`http://man7.org/linux/man-pages/man1/tcpdump.1.html`

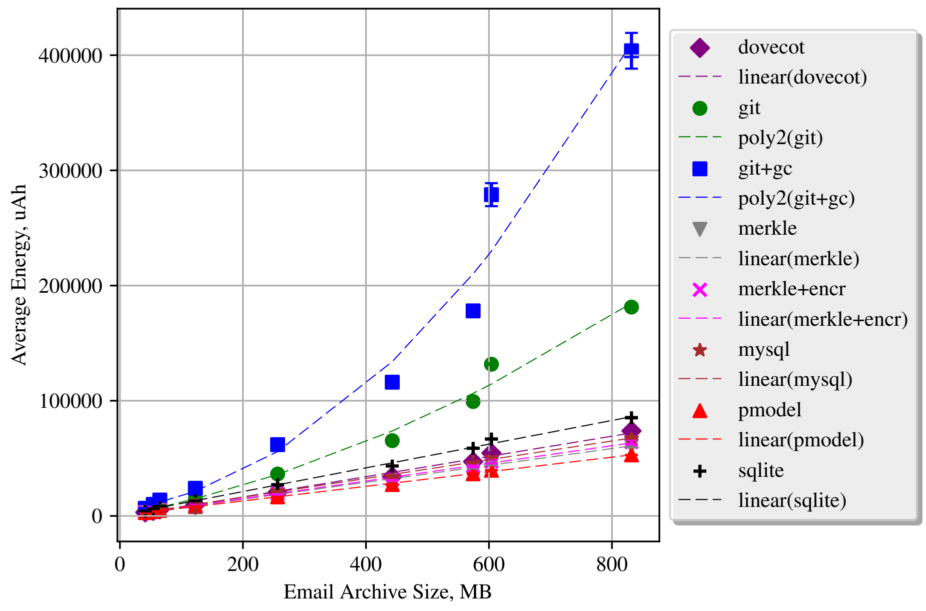


Figure 2.7: Message Append, energy use.

Table 2.2: Archive size (MB) for different store types.

N-Messages	250	500	750	1 000	2 000	3 000	4 000	5 000	6 000
Size(MB)	42.18	54.18	64.85	122.65	256.4	442.8	574.27	604.05	831.74
Pmodel	25.91	31.38	30.97	58.78	116.14	200.1	264.02	265.32	371.94
Dovecot	25.93	31.43	31.04	58.88	116.34	200.39	264.42	265.82	372.53
MySql	26.07	31.59	31.23	59.09	116.63	200.78	265	266	373
Sqlite	26.11	31.73	31.44	59.4	117.3	201.83	266.25	268.11	375.27
Git	27.89	39.07	48.1	89.07	236.23	470.02	743.16	1 013.4	1 449
GitGc	26.05	31.65	31.45	59.37	117.35	201.5	266.2	268.98	374.21
Merkle	25.93	31.43	31.04	58.88	116.33	200.39	264.41	265.81	372.53

concurrently. One explanation could be that the waiting on net-io and disk-io is replaced by the compression task, therefore reducing energy consumption on idling, which on the Raspberry Pi is about 250 mW. For comparison, under full load the energy consumption is about 400 mW.

If we take the Pmodel as the base then Merkle, Mysql, Dovecot, and Sqlite have approximately 14%, 27%, 39%, and 60% higher energy usage. It is not surprising that Merkle store energy usage is close to Pmodel one since the former essentially only adds SHA1 calculation as the overhead, which appears to be not significant. Without the source code analysis of Dovecot, we can only assume that it must be using more resources because of the user's account management or file locking in addition to the basic computational tasks that are described above. It is reasonable to expect that MySql and Sqlite use more energy resources due to the database management related processing on top of the basic tasks.

In contrast to the five described stores, Git and GitGc have substantially higher energy use. The reason for the higher energy becomes evident if we look at the size of the email

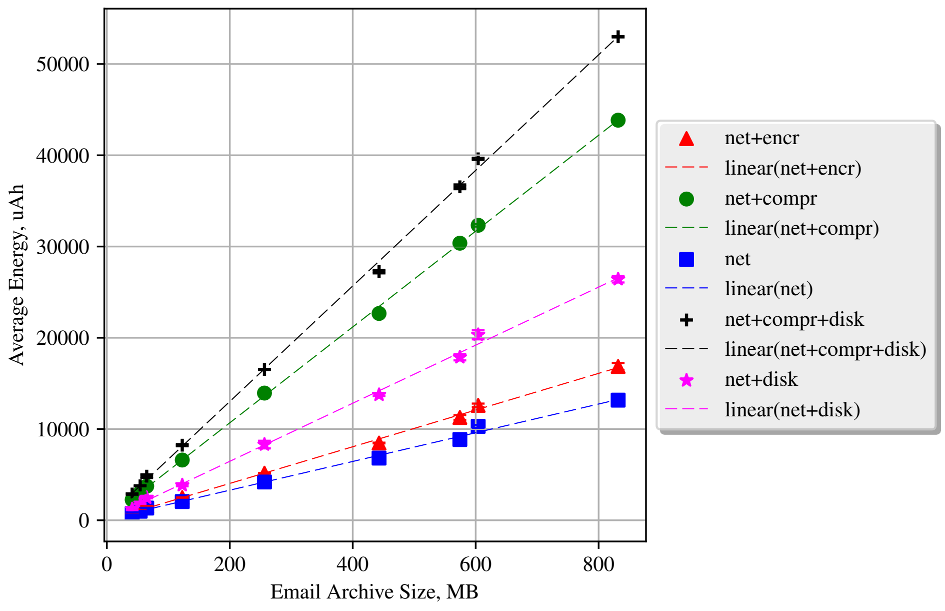


Figure 2.8: Average energy for net-io, disk-io, compression, and encryption tasks.

archive of appended messages for different stores, which is shown in Table 2.2. As can be seen, Git writes increasingly more data to the hard drive. The reason for this is the way Git maintains its history of changes. Each commit in Git is saved as the current state's snapshot of files, consisting of three object types: blob, tree, and commit. All objects are content-addressed via SHA1. Blob contains the content of the email. Tree contains the list of blobs as pairs of blobs or another tree SHA1 and its logical name, the list of messages in the Inbox in the case of email. Tree can contain other tree objects. Tree is conceptually the directory list. Commit contains SHA1 references to the current and parent's root tree and some additional metadata. If we add one message to the Git repository then the tree object may look like this, showing mode, object type, SHA1, and file name:

100644 blob 44677616312cf32ba8197b705c6947651dbcda 1

For simplicity, I assume that the file name is message's UID, 1 in this case. When another message is added a new tree object is created:

100644 blob a144677616312cf32ba8197b705c6947651dbcda 1

100644 blob 21640c373197d5558c05391bbc69f80a40347647 2

With each added message, a new tree object is created containing the list of all previous blobs plus the newly added blob. Previous tree objects persist on the hard drive. The size of each entry in the tree object is 52 bytes ($mode + type + SHA1$) plus the file name size. For simplicity, let us assume that the total entry size is 100 bytes. In this case, the size of all created trees for N messages is $(100 + 100 * N) * N/2$. For 6 000 messages the size is roughly 1.8 GB. Git compresses all objects, which is why the size in Table 2.2 is smaller - 1.449 GB.

One note on MySql archive size. While disk space overhead is small like in Pmodel etc., it only reflects MySql table size, and the overall database size is higher: 182.42, 193.78, 193.36, 249.4, 365.73, 535.28, 664.9, 668.99, 883.87 MB. MySql pre-allocates log

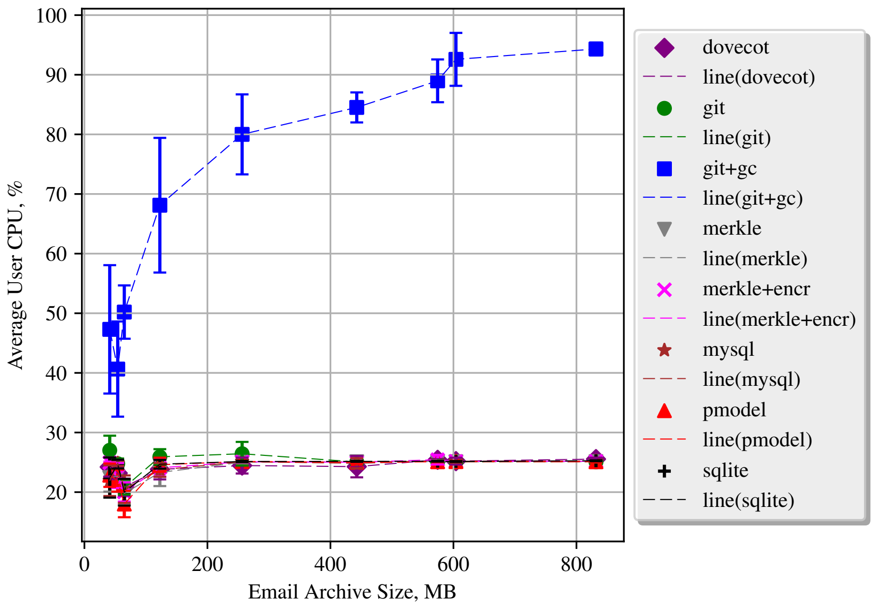


Figure 2.9: Message append, CPU usage.

files and system tablespace. While this process does not consume more energy, it does put higher constraints on the disk space resource requirement.

Why GitGc has even higher energy consumption than Git, even though the GitGc archive size is the same as Pmodel, Merkle, Dovecot, MySql, and Sqlite? Let us look at CPU usage by each store, shown in Figure 2.9. All stores except for GitGc have CPU usage around 25% while GitGc's CPU usage increases with the archive size. The reason is two-fold. First, the process of delta-compression and packing is computationally expensive. Second, even though the resulting archive size has a small overhead, each time a new pack is created it is written to a temporary pack first and when it is complete, the old pack is deleted. The new pack will be of larger size than the old one. Since we run GC every 250 messages, which is a relatively small incremental disk usage, the temporary disk overhead from Gc is roughly two-fold. Overall, except for the GitGc, the CPU usage is stable and the processing does not appear to be CPU-bound.

Let us look at other metrics. Figure 2.10 shows the latency measurements - the time it takes to append N messages to the store. Latency roughly follows the energy plots with similar store's ranking and trend-lines. Intuitively this makes sense - the more time it takes to run a task, assuming the same processing speed, the more energy it is going to take. Indeed, both energy and latency are linear functions of disk and net IO, and compression. In fact, if we plot Git's energy consumption depending on the resulting repository size after the messages were appended, rather than on the messages size being appended, then the trend-line is linear as shown in Figure 2.11.

Memory usage for different stores is shown in Figure 2.12. The ranking from best to worst is Dovecot, Pmodel, Merkle, Sqlite, MySql, Git, and GitGc. Not surprisingly, GitGc has the highest memory usage, which can probably be explained by the delta-compression and packing memory requirements. In Git and GitGc, I allocate the network buffer to receive the complete message, which could be up to 13 MB. Together with the program

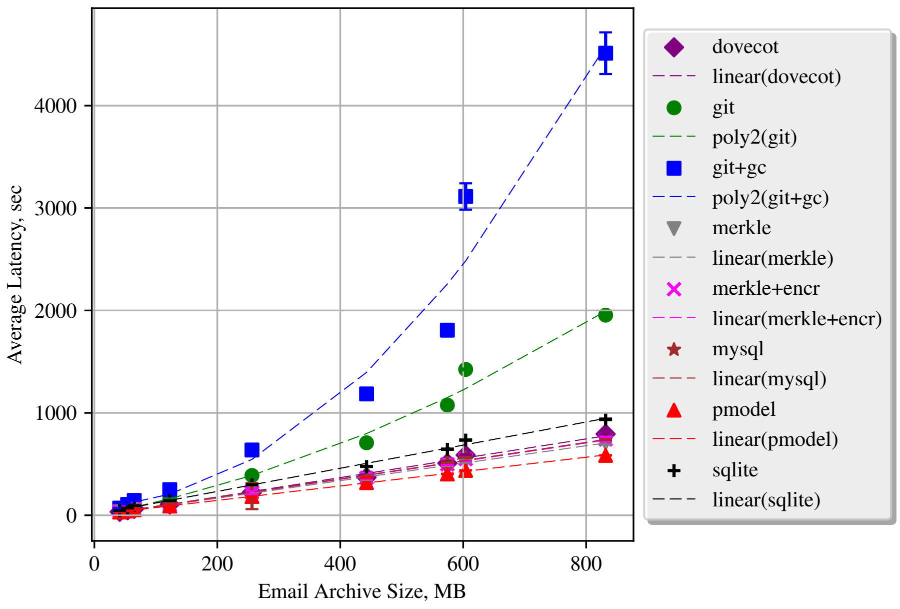


Figure 2.10: Message append, latency.

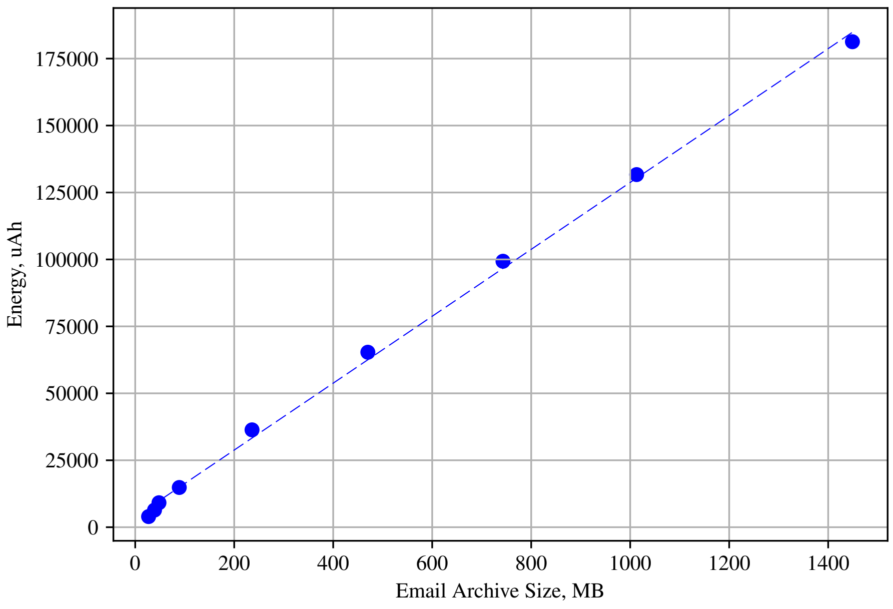


Figure 2.11: Git append energy use depending on the repository size.

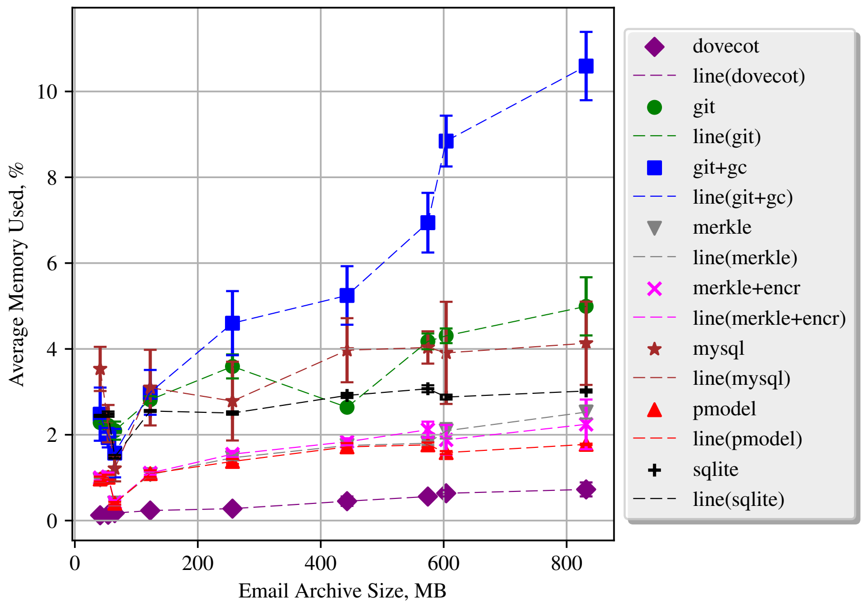


Figure 2.12: Message append, memory usage.

code of 7.7 MB this accounts for most of the memory used at a small email archive size. Memory usage increases with the archive size in Git because I cache the most recent tree object and Git could have a caching of its own. Similarly, in Pmodel and Merkle the initial memory usage is approximately equal to the pre-allocated 15 MB network buffer plus 1.64 MB program code. Sqlite and MySql have the same initial memory plus additional 5-15 MB for the database management. Similarly to Git, memory usage is increasing because I cached the message index. Overall memory usage, even for GitGc, is relatively low and processing does not appear to be memory-bound.

As I mentioned above, encryption (Merkle-encrypt) adds relatively small overhead and closely follows Merkle model in all metrics. Consequently, I exclude it from other tests except for FETCH to evaluate the decryption overhead.

I also look at the metrics of appending 100 messages for different stores to archives created in the previous append tests. The rationale is that the ideal store should have the same resource usage for the same appended size of messages regardless of the archive size to which the messages are appended: i.e., resource-wise cost of appending a message to the Inbox with 100 messages or with 100 000 should be the same. We can predict from the append message experiments above that Pmodel, Merkle, MySql, Sqlite, and Dovecot meet this criterion. The plots for appending 100 messages somewhat validate this expectation. Figures 2.13, 2.14, and 2.15 show Energy use, Latency, and CPU use respectively.

Indeed, we can see from the plots that energy use and latency are close to constant with CV for energy use averages of 0.02, 0.15, 0.06, 0.04, 0.09 and CV for latency averages of 0.02, 0.15, 0.06, 0.04, 0.09 for Pmodel, Dovecot, Sqlite, MySql, Merkle respectively. CV for Dovecot in both cases is higher - 0.15 and we can see a slight upward trend. In fact, trend-lines for all stores except GitGc and Sqlite are polynomial. The Sqlite trend-line is power and GitGc trend-line is upward linear; i.e., all consume resources not proportionally

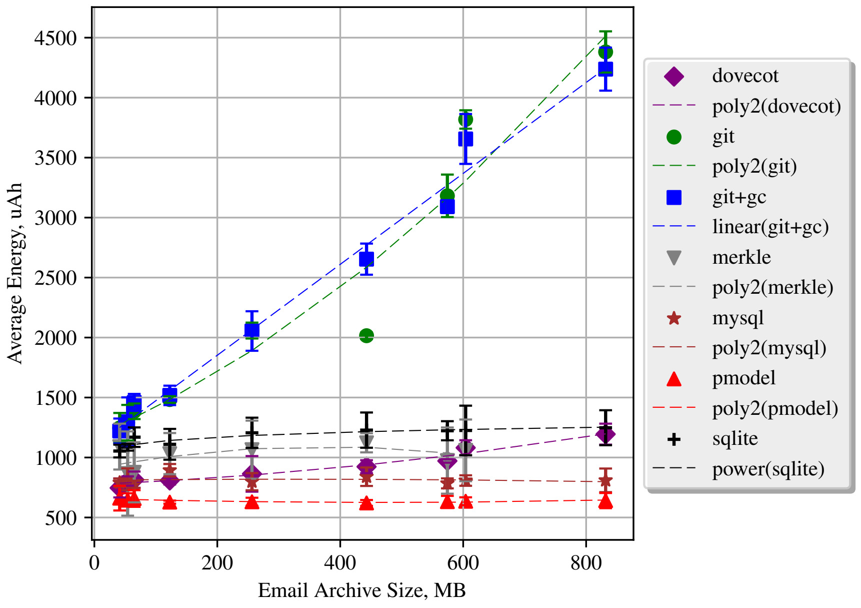


Figure 2.13: Append 100 messages, energy use.

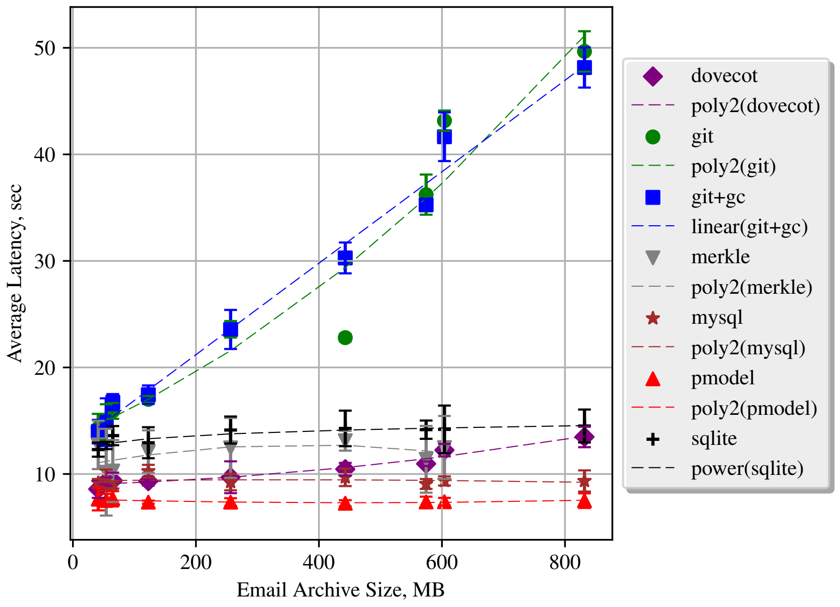


Figure 2.14: Append 100 messages, latency.

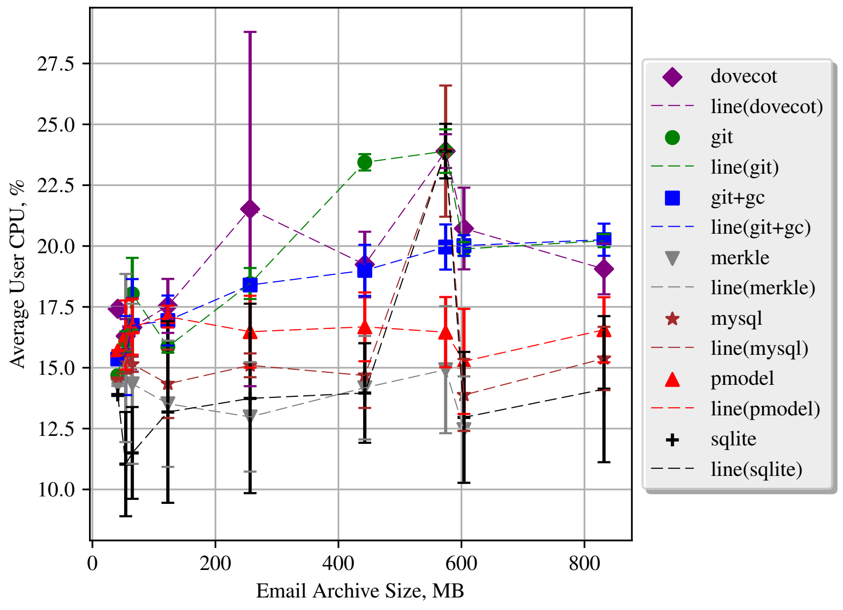


Figure 2.15: Append 100 messages, CPU usage.

to the archive size. This property is visually observable for Git, GitGc, and somewhat for Dovecot, but not so obvious for other stores. It might be a trend-line over-fitting issue. CPU plot, shown in Figure 2.15 has more variation in all five stores but generally is under 25% like in Figure 2.9 for Append messages.

Memory use is relatively flat across all stores as shown in Figure 2.16, with CV of 0.07, 0.29, 0.02, 0.03, 0.08, 0.08, and 0.04 for Pmodel, Dovecot, Sqlite, MySql, Git, GitGc, and Merkle respectively.

As expected, Git and GitGc have energy and latency increases at a higher archive size.

2.4.2 Fetching Messages

Figure 2.17 and 2.18 show average energy use and latency to fetch all messages from different store types and archives of different size. I use IMAP FETCH command to fetch messages. Best to worst ranking for energy use is Merkle, Pmodel, Sqlite, GitGc, Mysql, Git, and Dovecot. Latency has a similar ranking except for MySql and GitGc swapping places. In all cases, the trend-line is almost perfect linear with R^2 above 0.99. Dovecot performed the worst but then it must handle many aspects of a user's account management, like locking, whereas my implementation is a simple prototype with no concurrency support. It was a bit surprising that Merkle performed better than Pmodel. I did use a different SD card to run Merkle tests, consequently the difference can be explained by slightly better read performance of the SD card. Merkle and Pmodel have the best performance because there is a direct reference to each message via the message's ID with no index lookup; i.e., message's logical name corresponds to its order in the index. Sqlite and MySql access messages via individual select statements therefore there is the index lookup first. Sqlite fares better probably because its database is contained within a single file with no multiple file opens required. Git and GitGc access messages via the

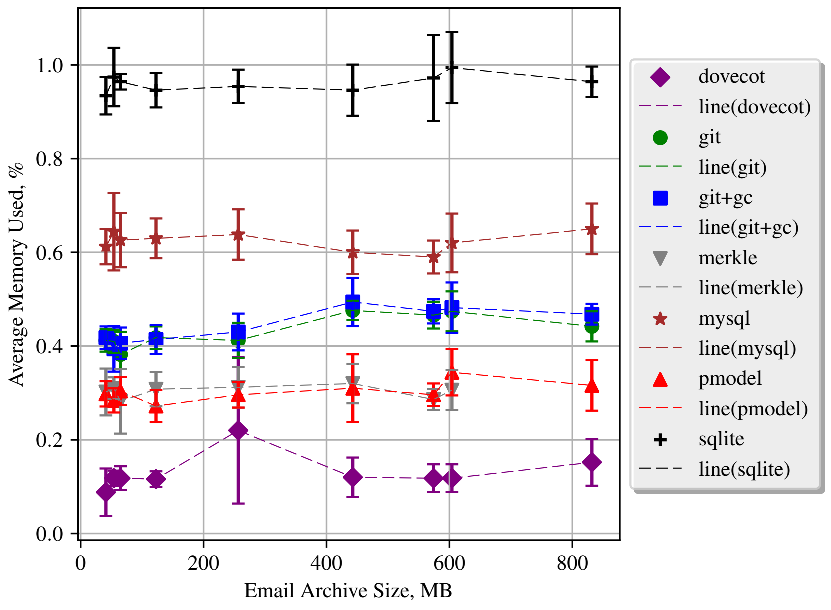


Figure 2.16: Append 100 messages, memory usage.

index tree object. GitGc performs better than Git because it only needs to open the single pack file, whereas Git has to open every message file.

Figure 2.19 shows CPU usage. There is an upward trend for all stores until the point corresponding to 5 000 (600 MB) archive messages and then it levels off. This is most likely caused by the variation of message properties within the archive, for instance some messages may require less or more computational resource for compression. All stores performed below the Dovecot's base line.

Figure 2.20 shows memory usage, which has even more variation. There is an upward trend for Git, Sqlite, GitGc, and MySql. This can probably be explained by increased caching as more messages are fetched. Pmodel, Merkle, and Dovecot have upward trend as well but well below the stores described above with Dovecot having slightly higher memory usage after the point corresponding to 4 000 (420 MB) archive messages.

Bandwidth usage in all cases is the same since once the message is read from the disk and uncompressed, it produces the same size regardless of the store type.

2.4.3 Archive Synchronization

Figures 2.21 and 2.22 show average energy and latency usage to synchronize local to remote archives for different store types and archives of different size. The remote archive has 100 unique messages appended to it with total size 8.54 MB. Best to worst ranking for energy use and latency is MySql, GitGc, Git, Dovecot, Mysql, Pmodel, and Sqlite. The trend-line in all cases except for Sqlite and Dovecot, which are linear, is second order polynomial with R^2 above 0.95. MySql's trend-line, while polynomial, appears to be almost perfectly horizontal. *Rsync* is used for Pmodel and Sqlite synchronization, Git's *fetch* command for Git and GitGc synchronization, Dovecot's *dsync* replication utility, and MySql is synchronized via the binary logs generated by MySql as part of the tables update.

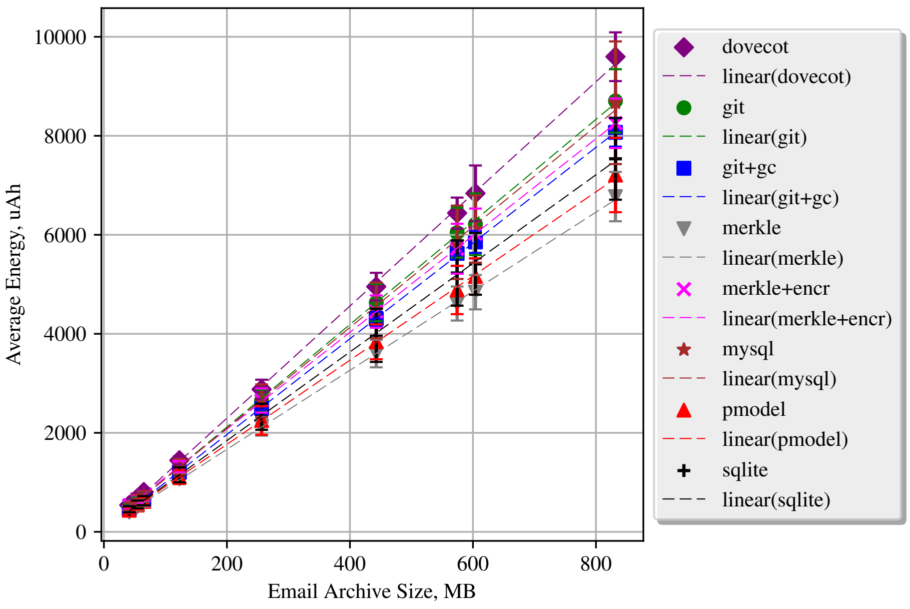


Figure 2.17: Fetch messages, energy used.

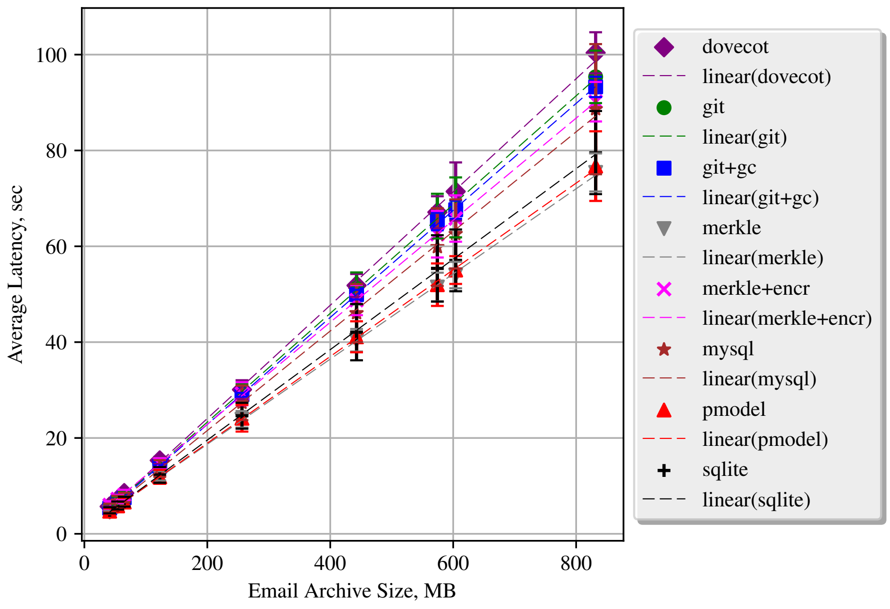


Figure 2.18: Fetch messages, latency.

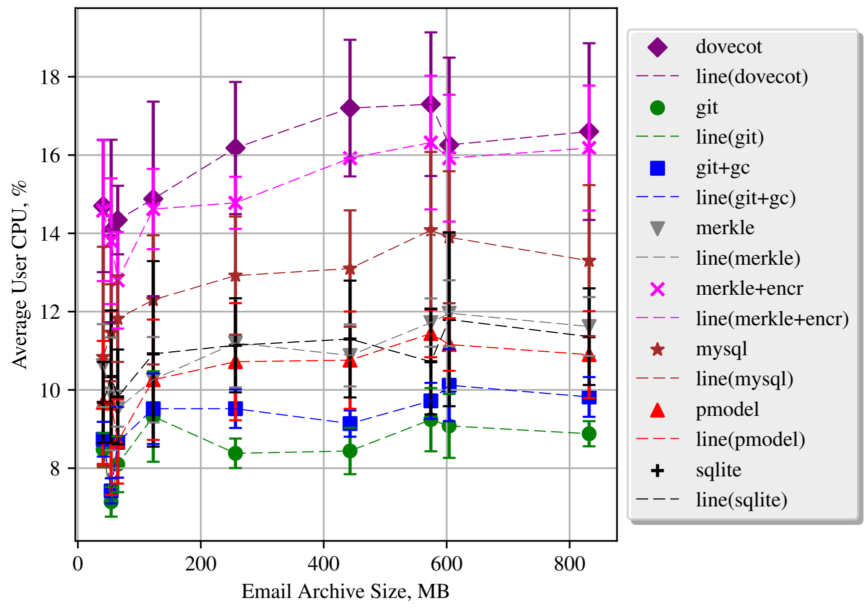


Figure 2.19: Fetch messages, CPU usage.

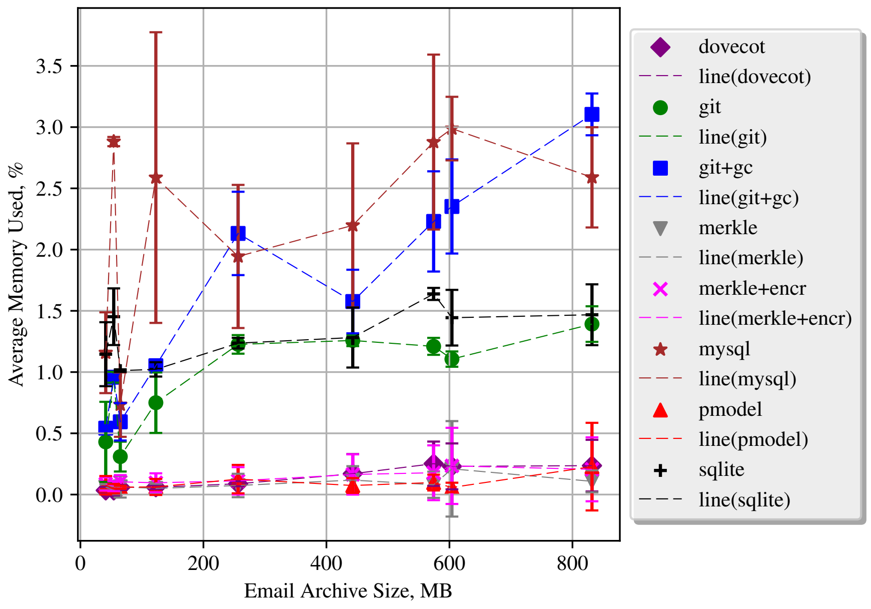


Figure 2.20: Fetch messages, memory usage.

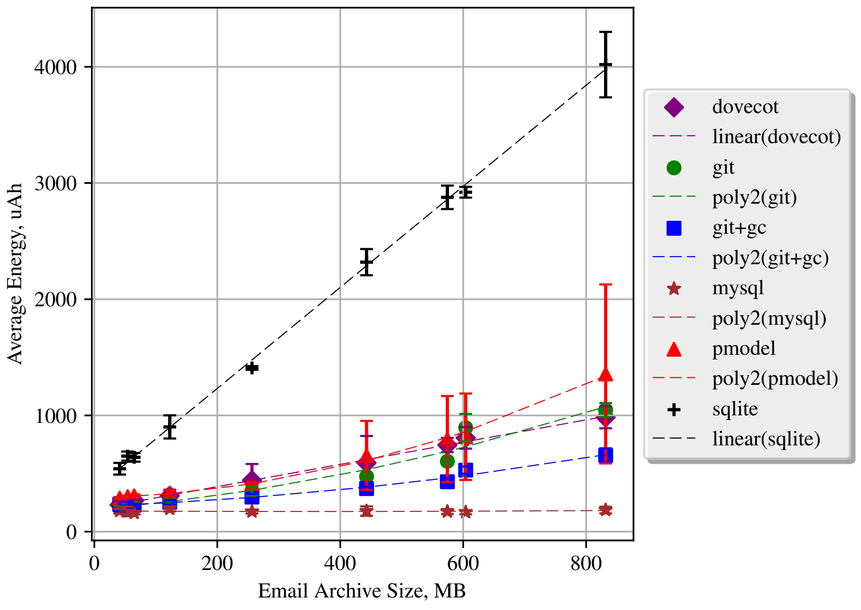


Figure 2.21: Sync messages, energy used.

Sqlite stores all its data in a single file. *Rsync* uses delta encoding and compression to transfer only the blocks that are changed which preserves the bandwidth. As can be seen from bandwidth use in Figure 2.25, there is a slight increase over 5 MB, which roughly corresponds to the compressed size of the added 100 messages; i.e., *rsync* only transfers changed blocks. But *rsync* has to re-assemble the blocks once they are received from the remote, which takes more CPU and memory resources as can be seen from Figures 2.23 and 2.24. On the other hand, MySql's binary logs simply tell the database engine what queries need to be executed on the local archive to synchronize it with the remote. This means MySql writes the same amount of data to the disk regardless of the archive size with no need to figure out the delta of the changes. Consequently, MySql's energy, latency, CPU, and memory are close to constant. *Rsync* in the case of Pmodel store needs to discover added files. Since the number of files increases with the larger archive, *rsync* needs more CPU and memory to scan through the list of files to discover the added ones. While Dovecot uses its own synchronization utility, like *rsync* it needs to discover added messages, which it does by comparing messages UID. The more messages there are in the email archive the more resources it takes to discover the changes. Intuitively, I expected Git and GitGc to have resource usage independent of archive size since Git uses Merkle hash tree to discover the divergence between the local and the remote archive. The problem is again in how Git maintains revisions via snapshots. As the number of messages in an archive grows, so does the number of entries in the tree object. Even though the number of delta messages that have to be synchronized is low, 100 in this case, the size of tree objects could be large.

I did not run synchronization test for the Merkle store. I could have used *rsync* with measurement results similar to Pmodel's one. Clearly, Git has better synchronization performance because it uses Merkle's hash tree as I mentioned above. But implementing the packing algorithm, required for synchronization, is a non-trivial effort, which I leave

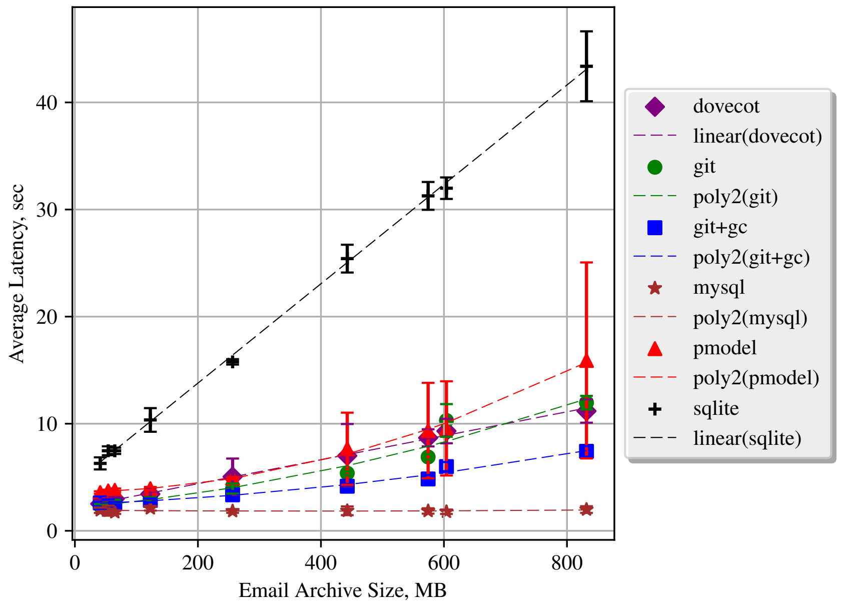


Figure 2.22: Sync messages, latency.

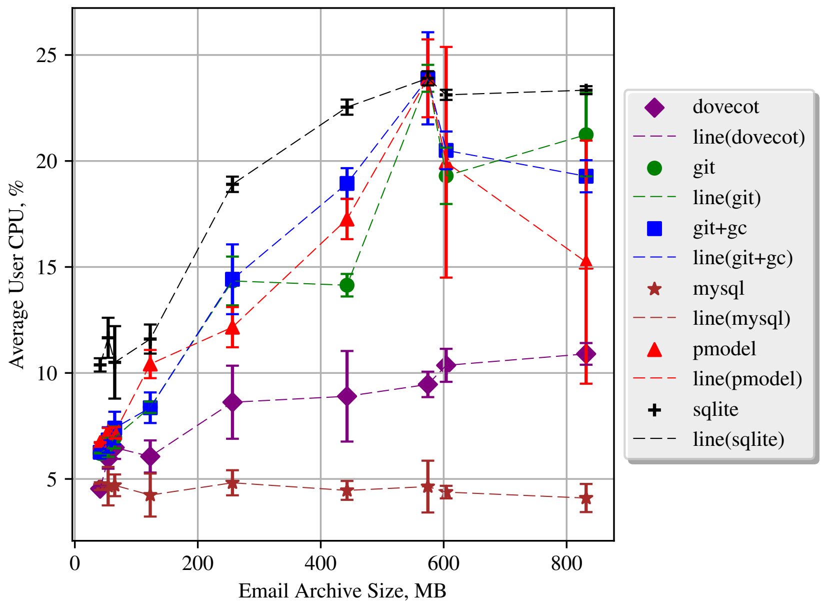


Figure 2.23: Sync messages, CPU used.

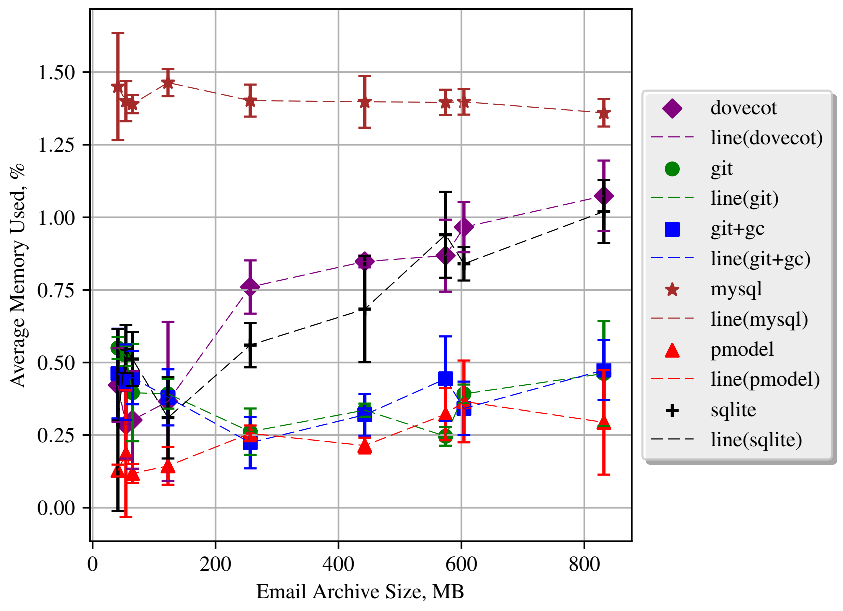


Figure 2.24: Sync messages, memory used.

for future work. Since the Merkle store overall has the same versions information as Git does, I expect it to produce the same pack file and therefore have a similar synchronization performance with Git.

2.4.4 Analysis

Evaluation shows that a VCS-like structured store can perform at least as well with respect to resources as the base Dovecot IMAP server. However, a VCS implementation can be disk space and CPU bound when the email messages are committed to disk. Git's snapshot implementation of revision maintenance has $C * N^2$ disk space, energy, and latency dependency. Git's Garbage Collection can substantially reduce disk space overhead at the expense of more energy, latency, CPU, and temporary disk space usage. MySql may double the disk space usage due to pre-allocation of log files. Access to messages is not affected by the type of the store. Evaluation of the synchronization protocol shows that except for MySql, which in this case has the best metrics due to pre-configured log files, Git has the best performance with respect to resources over Dovecot and *rsync*. This can be explained by the efficient Merkle hash tree detection of divergence between the local and the remote archive.

Evaluation further shows that the network-io, disk-io, compression, and encryption have a linear dependency of energy usage and latency on the bandwidth. Bandwidth can be reduced by compressing the data, transmitting binary data instead of base64-encoded data, and not sending duplicate attachments. IMAP extensions RFC 4978 [70] and RFC 3516 [113] support compression and binary data transmission. IMAP core RFC 3501 FETCH command[36] supports BODYSTRUCTURE extension data (optional) with the message's body MD5 but does not have an option of getting the attachment's MD5. A

[36]`https://tools.ietf.org/html/rfc3501#page-54`

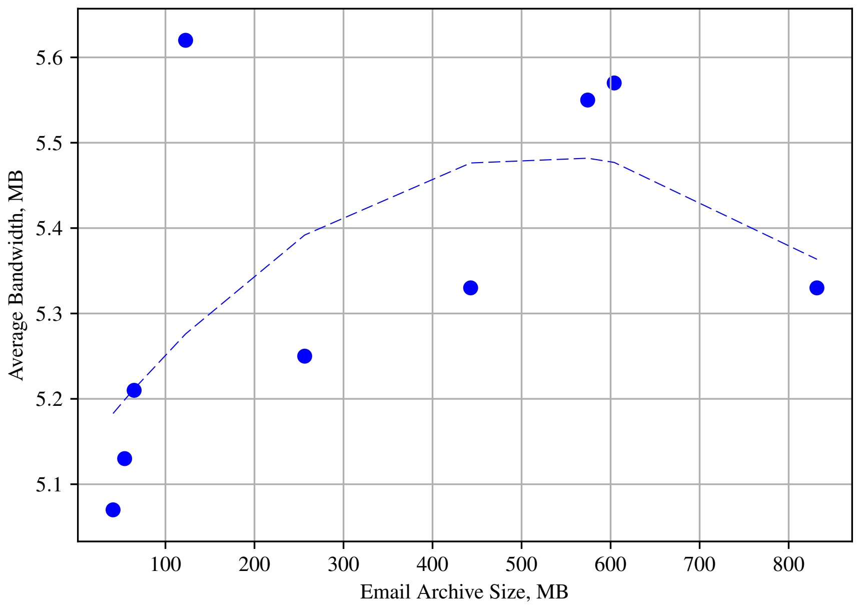


Figure 2.25: Sync messages, bandwidth used.

IMAP extension can be defined to support MIME parts identification via a hash like MD5 or SHA1. This will enable clients supporting this extension to choose whether an attachment needs to be downloaded or not. But IMAP protocol already has a rather complicated heuristic based on the mailbox's statistics and metadata to synchronize the client's cache to the server's database. Moreover, supporting version control in IMAP can further complicate the already extension crowded protocol.

2.5 Limitations

Below I identify some limitations of the evaluation:

- The evaluation considers the Raspberry Pi as representative of the resource-constrained device in IoT environment. This is a reasonable assumption considering Raspberry Pi affordability, popularity, ease of operation and its use by a IoT and distributed system researchers. Some example are the personal Databox in the security analysis of the distributed home system in Zhao et al. [158, p2], the Signpost DNS server in Rotsos et al. [125], or the multi-agent system with location aware and tracking system in Semwal and Nair [132].

- The base system IMAP server runs on the Raspberry Pi rather than on a centralized server platform. Evaluating a real email server is expensive in terms of cost and effort. Evaluating one of the publicly available email servers, for instance Gmail, is limited to latency measurement only, and is inaccurate due to the network latency. The IMAP server on Raspberry Pi emulates a dedicated personal email server running in the Cloud. This is not an unreasonable limitation considering that Raspberry Pi is an affordable way to host a dedicated personal email server and there is a commercial Nomx email server based on Raspberry Pi[37].

- I consider the three basic email functionalities of append, read, and synchronize executed on unstructured messages. Search is another popular function. It could be decomposed into reading the message, which is evaluated by the read function and the search algorithm. Search algorithm performance is implementation dependent, consequently evaluation results could be biased. We could consider the read evaluation as the worst case search scenario where all messages match some search criteria. Future work may consider evaluating Dovecot's search capability. The same argument of implementation dependency applies to the manipulation of unstructured rather than MIME parsed messages. MIME parsing enables access to email message parts and indirectly, if implemented, single storage. The latter can be evaluated from email statistics. We can evaluate Dovecot's MIME parsing in future work.

- Email back-end evaluation is limited to several implementations. The goal of the evaluation is to asses how resource-constrained devices handle basic email functions under a load while maintaining a history of changes, which is necessary in a distributed environment for efficient conflict resolution per CAP theorem. I suggest some common, albeit limited, back-end implementations such as open-source IMAP server (Dovecot), a relational database (MySql, Sqlite), NoSql (Git, GitGc), and naive VCS (Pmodel, Merkle). The evaluation of these back-ends provides a range of measurements, which allows establishment of low-high resource usage boundaries. Regardless of the back-end implementation, the tasks that affect the device resources are disk-io, network-io, compression, and encryption. Depending on the back-end implementation type, there is some overhead that adds to resource consumption. Consequently, this evaluation includes resource usage by the main tasks (io, etc.), which is constant across different implementations, and the range of some possible overhead. The measurements are done on a single Raspberry Pi device. These data are extrapolated to estimate energy consumption by a cluster of devices (Section 2.7). A more accurate result can be obtained by implementing a DFS connecting several Raspberry Pi devices or building a system on top of the distributed database, for instance Irminsule.

2.6 Availability

Existing centralized Internet Service Providers (ISP) provide high data availability to users. For instance, Microsoft, Amazon, and Google had downtime in 2017 of 740, 205, and 11 minutes respectively, which in average is 121.6, 33.7, and 1.8 seconds downtime per day[38]. Google achieves high availability by replicating the data between the clusters of geographically distributed servers like in the Spanner globally-distributed database [42]. Users will not adapt an architecture which is not comparable in availability to centralized services even if it promises better privacy. Indeed, what good does privacy do if we can not access our email, or even worse - lose it? Users can tolerate some delay in posting

[37]`https://www.nomx.com`, `http://www.bbc.com/news/technology-38934822`

[38]`https://www.theinformation.com/articles/how-aws-stacks-up-against-rivals-on-downtime`

or accessing their messages but not their loss. In a decentralized email architecture, like P2P, availability is accomplished through replication as shown in Section 2.1.3. Some research on P2P email includes availability analysis. Zhao et al. [159] analyze the number of required email replicas depending on the peer's down probability. They show that the number of required replicas is less than 5 with 0.1 down probability and is about 10-18 with 0.5 down probability depending on desired availability of 99%-99.99%. Generally, the number of replicas increases slowly up to 20 until it reaches down probability of 0.7 and then increases exponentially. Mezo et al. in DMS [106] show a higher number of replicas are required, with 25 when the down probability is 0.1 to 50 with the down probability of 0.5. Zhao runs simulations in a network consisting of five communities with 20 nodes each. On the other hand Mezo analyzes a network of 1 000 nodes and therefore has higher churn rate. In addition, Mezo considers users who briefly login into the system to read email as high down time. But this does not necessarily mean that the user's device is not connected to the Internet. But the analysis of availability also by Mezo et al. [107] in HMail, built on DMS, show more optimistic availability numbers. Though visually difficult to discern from the figure, the number of replicas is under 25 in the case of 0.5 down probability and under 5 for down probability below 0.3. HMail simulation also runs on a 1 000 node network but it implements a higher level of hierarchical organization than DMS, which improves availability. Research into P2P storage system by Song et al. [136] suggests that the number of required replicas for 99.99% availability is three when the files are placed on peers with high availability. Interestingly, both DMS and HMail are based on the same research in [136]. Bolosky et al. in [29] show that high availability can be reached with up to five replicas in a serverless DFS in corporate environment. The research into decentralized mobile OSN MobiTribe shows that nearly 100% of content availability can be achieved with two replicas if the devices are carefully grouped into tribes based on history of device availability patterns of devices, Thilakarathna and Petander [142]. In another P2P OSN, SOUPS [88], Koll et al. suggest that six replicas are required to match a cloud-based service availability. Sharma et al. [133] explore the trade-offs between redundancy, data placement, and availability in the friend-to-friend storage systems. These authors are looking to find the minimal number of friends that provide maximal benefit with minimal replication. They define two optimization parameters. First is achievable coverage (AC) which is the fraction of the total time for which a node n is able to get data availability by storing data at all its friends, out of the total time for which it is seeking data availability. Second is degree of criticality (DC) which is the fraction of time that the coverage is provided by a unique only up neighbor out of the total time that the node n is online. Experiments are carried out on the trace-driven simulations of IM traces. Results show that reasonably good availability can be achieved with 10 friends.

Depending on device availability, architecture, and the analysis, the number of required replicas is in the 2-50 range. Perera et al. in the IoT Databox [117] assume SOUPS provides reasonable availability with six replicas. The authors further suggest that the downtime of a Databox can be reduced by powering the device with backup batteries. Moreover, they recommend supplementing the broadband connection with GPRS/Edge dongle or connecting to the Internet via neighbors' shared broadband link. The email architecture I am suggesting, just like Databox, runs in the home IoT environment. I am assuming the 10 replicas as the upper boundary to achieve high availability in the decentralized IoT email architecture. Achieving the goal of 10 replicas in the home IoT system is a reasonable assumption. According to Pew Research, 90% of U.S. households contain at least one of these devices (smartphone, desktop/laptop computer, tablet or streaming media device), with the typical (median) American household containing five such devices. And nearly one-in-five American households (18%) are "hyper-connected" meaning they contain 10 or more of these devices[39]. This report does not take into account the availability of a network router. According to Pew Research, 73% of US adults had broadband Internet in 2016[40] with 50% of the population having high speed connection of 50MBps and 95% the speed of 10MBps[41]. Even if a user does not have enough combined devices to provide 10 replicas for email messages, she can add affordable Raspberry Pi devices to her home environment or pay for virtual Cloud storage. It is also possible that in some cases two replicas, like in MobiTribe, is sufficient to achieve high availability. There are two reasons for this. First, it is reasonable to assume that home IoT devices do not have high churn rate if any. Indeed, a home router is a stationary device and does not go through on and off power cycles. It is continuously connected to the Internet as long as there is no power outage (backup battery addresses this issue) and ISP does not experience downtime. According to U.S. Energy Information Administration, customers in 2015 experienced in average about 200 minutes of the power outage or 32.8 seconds per day[42]. There are no reliable data available on an Internet provider's downtime. We can use other Cloud providers downtime as an indication of Internet availability. Cagnaire et al. in [62] report the average of 7.738 hours Cloud services were unavailable per year in 2012-2013 or 99.91% availability. If the Internet is not available, a user still has her smartphone to send and receive email and replicate the messages to other home IoT devices and conventional computers via WiFi. Naturally, if Cell service is down as well, then the user experiences a complete email block-out. In this case, the sender stores the outgoing message in her Outbox until the receiver is online again. As long as the user's globally accessible devices are discoverable, this use case is not different from centralized architecture availability. Google's approach to data availability is "Availability of data is more important the availability of access. If a system is down its not the end of the world. If data is lost, it is."[43]. In that regard, having all email replicas located in a user's home may result in the user losing all her email messages. Indeed, in case of a fire, a user may have to evacuate the house in a rush leaving behind all her possessions, including her smartphone; i.e., the home is a single point of failure. To mitigate this problem, a user may share devices with her close friends and family accomplishing high availability to both her and the participating party. In this case some replicas reside at a different location eliminating the single point of failure issue. The choice of other users to share the device with or the cluster of user's devices can be based on a social analysis of the user's data. Chapter 3 presents a novel way of performing this analysis based on the email attachments shared between users and Section 3.7 applies this analysis to estimate the energy savings from clustering users into groups sharing the most email attachments in common. Section 2.7 includes the evaluation of the energy cost of a decentralized email architecture with 2-10 email replicas.

[39]http://www.pewresearch.org/fact-tank/2017/05/25/a-third-of-americans-live-in-a-household-with-three-or-more-smartphones/

[40]http://www.pewinternet.org/fact-sheet/internet-broadband/

[41]https://www.broadbandmap.gov/download/Broadband%20Availability%20in%20Rural%20vs%20Urban%20Areas.pdf

[42]https://www.eia.gov/todayinenergy/detail.php?id=27892

[43]http://highscalability.com/blog/2014/2/3/how-google-backs-up-the-internet-along-with-exabytes-of-othe.html

We can conclude that high availability can be achieved in decentralized email architecture with up to 10 replicas of the email messages if some of the replicas are located outside of the user's home. It is conceivable that with the proliferation of IoT technology, a user may own devices that are geographically distributed consequently providing high-level availability without sharing the devices with other users. But an early adapter of decentralized email or other similar social applications has to share resources with other trusted peers, like friends or family, or use the Cloud backup service.

2.7 Energy cost evaluation

Driven by social media and video streaming demand, the on-line universe carries a large amount of data that needs to be stored somewhere, e.g. the data centers. Cisco [12] estimates that by 2020 global data center's IP traffic will reach 15.3 zettabytes (ZB) and the data stored will reach 915 exabytes (EB). It is interesting that the amount of data stored on devices will be five times higher than the amount stored in data centers at 5.3 ZB. The amount of energy consumed by data centers will triple in the next decade. It presently constitutes 3% of global energy usage and contributes 2% to green house gas emissions, Bawden [22]. To put things in perspective, 416.2 terawatt hours of electricity that data centers used in 2015 is significantly higher than the UK total consumption of 300 terawatt hours, and the carbon footprint is on the same level as the airline industry. To control the carbon footprint, companies increased their use of renewable energy; for instance, in 2016 Microsoft announced its largest wind purchase to date [13]. Google used machine learning AI to cut its data centers energy use by 15%, Vaughan [147]. To reduce cooling cost, which contributes 40% to the overall energy cost, Facebook opened some data centers in Sweden, 70 miles from Arctic Circle, Bawden [22]. Considering that data centers have a significant impact on energy demand, it is reasonable to ask how will the proposed decentralized architecture compare to the centralized data center in terms of energy cost.

Companies owning data centers generally do not release their detailed energy costs. But, as part of the green initiative, in 2011 Google published a white paper outlining an email energy cost comparison of cloud data center versus local server [9]. Google estimates that their per user email annual energy usage is less than 2.2 kWh.

For the purpose of energy cost evaluation of the proposed architecture I make the following assumptions:

- A user receives 250 new emails per day. The Radicati email statistics report [120] estimates that by 2019 the average number of emails send and received per user per day is 246.5.

- I assume that IoT home devices include one router with a globally accessible address connected to the Internet and one to nine other devices connected to the router and each other via home WiFi. A new email message is first appended to the router and then distributed to other devices via WiFi. All IoT devices are comparable in their resource usage with Raspberry Pi.

- The email back-end implementation is GitGc with the compression and encryption. The average energy consumed by appending 250 email messages to the back-end is 6631 μAh (Figure 2.7). To account for encryption, which adds a small overhead (Figure 2.8), I round up the energy consumption to $A = 7\ 000\ \mu Ah$.

- Fetching 250 email messages consumes on average $F = 458\ \mu Ah$ (Figure 2.17).
- I assume synchronization of email or replication consists of reading all new compressed and encrypted messages from one device, transmitting them to another device, and receiving and saving all received messages on that device. I ran additional tests to evaluate the energy usage of the replication steps with the Raspberry Pi connected to a residential WiFi. I estimated this process to consume about $S = 6\ 000\ \mu Ah$.
- To account for the energy cost of sending messages to an email list, I assume that each email message is sent to five recipients. This is based on an average recipient list size of 7.07 and 2.14 in Enron and private archive, respectively. The energy cost to send a message to five recipients is $R = 3\ 000 * 5 = 15\ 000\ \mu Ah$. The cost of sending the message via WiFi is 3 000 μAh taken from the above estimate.
- I use Power Usage Effectiveness (PUE)[44] of 2.5 (highest estimate) in my analysis.
- The number of backups is N, where N is 1-9. This means there are 2-10 replicas of the email messages.

Per assumptions above, the total daily energy consumption E in Ah is

$$E = PUE * (A + F + R + S * N)/1\ 000\ 000 \tag{2.1}$$

Since the nominal Raspberry Pi voltage is 5.1 V, the annual energy Y consumption in Wh is

$$Y = 5.1 * 365 * E = 5.1 * 365 * PUE * (A + F + R + S * N)/1\ 000\ 000 \tag{2.2}$$

or

$$Y = 5.1*365*2.5*(7\ 000+458+15\ 000+6\ 000*N)/1\ 000\ 000 = 4.65375*(22.458+6*N) \tag{2.3}$$

Annual energy consumption using above formula is 160.36 Wh for two replicas and 355.81 Wh for ten replicas. This estimate is lower than Google's estimate of 2.2 kWh. Note however, that the annual Raspberry Pi 3 energy consumption, depending on the load, is $10(idle) - 31(100\%load)\ kWh$[45]. Consequently, if an IoT device is under-utilized, then energy consumption is substantially higher than Google's estimate. One more note is that 2.2 kWh Google's estimate is six years old. Shehabi et al. in [19, pES-2] show US Data Center total electricity use at the 2010 energy efficiency level and current various strategy efficiency level estimates. At the 2010 level, total energy usage in 2017 is about 150 billion kWh, and with best practices it is 40 billion kWh. Even if we roughly estimate a four-fold improvement in efficiency then Google's estimate is $550Wh$, which is 1.5-3.4 times higher than the projected decentralized email energy usage assuming it is reasonably utilized. We can find the maximum number of replicas at which the energy cost in the decentralized email architecture is the same as the Google's estimate. This number of replicas is 16, which means we can achieve a high level of availability at an energy cost at least as good as in the centralized system.

[44] `https://en.wikipedia.org/wiki/Power_usage_effectiveness`

[45] `https://raspberrypi.stackexchange.com/questions/5033/`

2.8 Summary

I presented a high-level decentralized email architecture that takes advantage of an IoT smart-home environment running resource-constrained computing devices with a publicly accessible address. The architecture maintains a full email history of changes, which is important in present day computing environment where intermittently connected to Internet multiple devices, like smartphone, tablet, etc. are accessing and making changes to the email archive and users are unintentionally deleting or misfiling an important email due to email overload. The proposed architecture replaces legacy IMAP and SMTP protocols with a synchronization protocol, which relies on Merkle hash tree to identify divergence between the local and remote archive. I presented a detailed evaluation of latency, CPU, bandwidth, energy, memory, and disk usage for various email back-end stores with and without the revision control. I also evaluated the energy usage of the decentralized architecture. Evaluation shows that resource-constrained devices, like Raspberry Pi, are capable of supporting email architecture with revision-controlled archives and that this architecture can perform at least as well as the conventional IMAP server and from the energy usage prospective is an efficient alternative to the centralized architecture.

While energy cost evaluation considers a set of backup devices owned by one user, I also suggest that socially connected users may share their devices. Section 2.2 reviews use cases of socially connected users sharing their devices. Section 2.6 discusses how sharing devices with friends and family can increase a system's availability. Sharing devices can also optimize the backup strategy of the email messages and consequently reduce the energy cost. The idea is that the email of socially connected users contains duplicate data such as email attachments. Consequently, we can reduce the number of required email attachment's backups since within a group of socially connected users the number of required email attachment's backups is intrinsically met. Chapter 3 shows a novel way of extracting a social network from the email dataset to discover groups of users sharing the most email attachments and how this information can be used to optimize backups and consequently energy cost. It also demonstrates how the same analysis can be applied to the dataset crawled from photo-sharing Flickr social network.

Emerging IoT technology is a promising platform not just for email but for the overall unified messaging architecture where data is put back under user's control. I only touched on a few aspects of this architecture but I hope it will encourage more research in this area. Future research can focus on an optimal data structure to maintain email revisions. Details of a synchronization protocol have to be examined to decide how to prioritize MIME parts download so that bandwidth-limited devices can have an email preview without downloading the whole message. Connecting peers is an area of much research with one thought being to use existing email providers as the DNS server. And, last but not least, user privacy in an IoT environment is a substantial research subject.

Chapter 3

Social network analysis of the email and Flickr datasets

3.1 Background

Chapter 2 discusses high level decentralized email architecture, evaluates if a resource-constrained IoT device like Raspberry Pi can support this architecture, and estimates the energy cost of the architecture. Energy cost is estimated by aggregating energy consumption from the basic email functions of appending and fetching email message and replicating email messages to 1-10 backup devices. It is assumed that the backup devices are part of the smart home IoT environment. Having up to 10 email backups or replicas is sufficient in terms of the availability as discussed in Section 2.6. This only holds true as long as there are no catastrophic events, like a fire, that may destroy all email backups, including mobile devices like a smartphone. To resolve this issue one of the replicas can be located in virtual storage. This comes at a cost to a user, either monetary or in terms of privacy. Another option is to use devices of trusted peers, for instance friends and family, as in Friendstore [144]. The question then is who should be selected from the user's trusted peers with whom to share the devices and support the backup requirements. Should it be a friend, a sibling, or someone else? The email dataset of participating users can provide answers to this question. As was mentioned in the Introduction and Section 2.2, a large proportion of the data (81-91%) in the analyzed email datasets are email attachments. Moreover, duplicate attachments use 27-34% of email message data. This simply means that users have identical email attachments in their email archives. Indeed, if a user sends an email with an attachment to a few recipients then both the sender and all recipients will have the same attachment in their mailboxes. Consequently, we can use social information contained in the email dataset to infer interactions between users. In this case we are interested in how users relate to each other through the sharing of attachments. Examining social information about users is commonly used in architectural decision making and optimization. For instance, a decentralized OSN Safebook [47] leverages trust between users for mirroring a user's data or profile data routing by peers that trust one another in a social network. Tribler [119] extends the popular P2P file sharing BitTorent system by building a social overlay on top of it by connecting people with similar tastes, known as taste buddies. Using collaborative downloading of taste buddies improves BitTorent performance by yielding significant speedup. Likewise, we can use social information inherent in a email dataset to optimize the backup of email messages by minimizing the amount of data that needs to be backed up. Optimization of a backup strategy is important not just from a user's perspective in order to reduce her energy cost, but it is also important from a societal perspective. As shown in Section 2.7, data centers have a substantial impact on energy demand and on carbon footprint; reducing energy usage makes a positive impact on the environment. In the rest of the Chapter 3, I propose a novel way of extracting a social network from email attachments shared between users, analyze the extracted network, and demonstrate how this analysis can be used to optimize email backups. In addition, I suggest that the same kind of analysis can be generalized and applied to OSN. I demonstrate this approach on the crawled dataset from the photo-sharing Flickr social network.

3.1.1 Concepts

In this section I provide an explanation of some centrality measures that reflect importance of nodes in social network and classification and clustering algorithms.

Social Network Analysis (SNA) is the process of investigating social structures through the use of networks and graph theory. It characterizes networked structures in terms of nodes (individual actors, people, or things within the network) and the ties, edges, or links (relationships or interactions) that connect them [16]. Before SNA can be applied, the network, which is the subject of the analysis, has to be extracted. Typically, in the context of the email archive, the extracted network reflects the frequency of communication between users. Once the network is extracted, various statistics and methodologies can be used for the analysis. In my research I used centrality measures of degree, betweenness, closeness, and eigenvector. The choice is in no way exclusive. These measures are commonly used in SNA to identify the most central nodes in the network and allow us to gain some basic knowledge about the shared attachments network. In addition, I use the k-nearest neighbor classification and k-means clustering algorithms. The choice is more pragmatic in that these methodologies may identify groups sharing the same attachments and consequently be applied to storage and bandwidth optimization of the decentralized architecture.

Degree Centrality[1] is historically the first and conceptually the simplest centrality metric. It is defined as the number of links incident upon a node (i.e., the number of ties that a node has). The degree can be interpreted in terms of the immediate risk of a node for catching whatever is flowing through the network (such as a virus, or some information). In the case of a directed network (where ties have direction), we usually define two separate measures of degree centrality, namely indegree and outdegree. Accordingly, indegree is a count of the number of ties directed to the node and outdegree is the number of ties that the node directs to others. When ties are associated to some positive aspects such as friendship or collaboration, indegree is often interpreted as a form of popularity, and outdegree as gregariousness.

Closeness Centrality[2] of a node is a measure of centrality in a network, calculated as the sum of the length of the shortest paths between the node and all other nodes in the graph. Thus, the more central a node is, the closer it is to all other nodes.

[1]`https://en.wikipedia.org/wiki/Degree_(graph_theory)`
[2]`https://en.wikipedia.org/wiki/Closeness_centrality`

Betweenness Centrality[3] is a measure of centrality in a graph based on shortest paths. For every pair of vertices in a connected graph, there exists at least one shortest path between the vertices such that either the number of edges that the path passes through (for unweighted graphs) or the sum of the weights of the edges (for weighted graphs) is minimized. Betweenness centrality for each vertex is the number of these shortest paths that pass through the vertex.
Eigenvector Centrality[4] is a measure of the influence of a node in a network. It assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes.
Jaccard index, also known as Intersection over Union, and the Jaccard similarity coefficient, is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. The Jaccard distance, which measures dissimilarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union.
K-Nearest Neighbor[5] is a non-parametric method used for classification. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor. K-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.
K-means Clustering[6] is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

3.1.2 Overview

There is an extensive body of research on SNA based on email communication. The subject of the research can cover multiple topics such as relation discovery in Shetty and Adibi [134], software project activity in Bird et al. [27], group inference in Yelupula and Ramaswamy [155], hierarchy detection in Duczynski and Yin [103], crisis analysis and prediction in Diesner et al. [51], topic and role discovery in McCallum et al. [100], text analysis of social values in Zhou et al. [161], fraud detection in Tang et al. [141], information extraction and search in Laclavík and Šeleng [92], classification in Wang et al. [151], and SPAM detection in Lam and Yeung [93]. Typically, the analyzed network reflects either communication between users or a relationship between the email and the information found in the email's header and the body. In the former, the nodes represent

[3] `https://en.wikipedia.org/wiki/Betweenness_centrality`
[4] `https://en.wikipedia.org/wiki/Eigenvector_centrality`
[5] `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`
[6] `https://en.wikipedia.org/wiki/K-means_clustering`

Table 3.1: Attachment's statistics, with (SD/CV) where applicable.

	Enron	Friends & Family
Archive size	23.83 GB	68.17 GB
Avg. Size of Attachments in Mailbox	91.21% (9.18/0.1)	81.93% (18.65/0.23)
Avg. Duplicate Attachments in Mailbox	14.9% (12.03/0.81)	13.84% (8.78/0.63)
Total Duplicate Attachments	34.8%	27.22%
Total Shared Attachments	29.87%	25.43%
Avg. Messages w/Attachments in Mailbox	24.60% (11.69/0.48)	14.26% (6.94/0.49)
Type of Attachments in all Mailboxes	Docs: 92.88% Multimedia: 5.73% Other: 1.39%	Docs: 17.02% Multimedia: 79.2% Other: 3.78%

users found in the *From, To, Cc*, and *Bcc* email header fields, and edges, either directed or undirected, represent the relationship between the sender (*From*) and the recipient (*To, Cc, Bcc*), with the weight reflecting the frequency of communication between the two, for instance as in Diesner et al. [51]. In the latter, the nodes represent the email and entities extracted from the email like people, phone number, email addresses, etc., and the edges represent co-occurrence of named entities within the email or its parts, including converted to text attachments, for instance as in Laclavík and Šeleng [92].

I analyzed private email archives of friends and family and the Enron email corpus. Statistics are shown in Table 3.1. While the average number of messages with attachments is only 14.26% (private) and 24.60% (Enron), the average disk space that they use is high at 81.93% (private) and 91.21% (Enron). Another observation is that 92.88% of attachments in the Enron's corpus are documents, while 79.2% of attachments in the private archive are Multimedia (Audio, Video, Image). Therefore, generally most of the data is discarded in SNA based on the communication network. While a multidimensional network based on entities extracts textual content of attachments, it ignores images, which may represent a substantial part of the email message. Besides making a quantitative contribution to SNA, attachments may have a qualitative property as well by representing the "intimacy" manifestation of the relationship strength. Granovetter defined the strength of a tie in his seminal paper "The strength of weak ties" as "The strength of a tie is a (probably linear) combination of the amount of time, the emotional intensity, the intimacy (mutual confiding), and the reciprocal services which characterize the tie" [67, p3]. Gilbert and Karahalios in [65, p6] show that "intimacy dimension" makes the largest contribution of 32.8% to the tie strength prediction model based on social media. Indeed, it is plausible to assume that sharing, for instance, a picture of an interesting event or an important document might indicate a higher level of trust and a close relationship between two people.

I am proposing to extract a social network based on email attachments shared between user accounts, constructing a one-mode projection of bipartite graph as in Wang et al. [150]. I view the attachment as a "virtual event" attended by users sharing the attachment. This is a variation of similar sentiment described in Gupte and Eliassi-Rad [71, p8]: "Each email is an event and all the people copied on that email - i.e., the sender (*From*) and the receivers (*To, Cc, Bcc*) are included in that event". Indeed, if Alice sends pictures of a New Year party to Bob then Bob "virtually" shares some, possibly the most exciting, a part of the experience with Alice. Likewise, if Bob sends some important document to Alice then Alice collaborates on the document with Bob via the "virtual" meeting. We can then extract the social network with nodes representing users and edges representing the attachments shared between the users. The way I infer the sharing of attachments is via the SHA1 of the attachment's content. This can be accomplished by parsing each email message of each user's email archive into its MIME parts, aggregating SHA1 of attachments by the user, and finding common attachments between users. Identical attachments have the same SHA1. But not every attachment represents a quality social relationship between users. For instance, some attachments could be a part of the bulk-email, making every user connected. Other cases include common logos, signatures, or Internet trends. Indeed, an intra-company's email with the company's logo attached will make all users connected. A methodology should be used to filter out attachments that are not meaningful for analysis. I hypothesize that a social network extracted from shared attachments will provide more insight into the relationships between people.

The contribution of this chapter are the following:

- I propose to extract the Social Network via shared email attachments (Section 3.2, 3.4).
- I propose a methodology of filtering out non-meaningful attachments (Section 3.2, 3.4).
- I extract the communication and shared attachments networks from Enron email corpus and private email archives. I then analyze degree, betweenness, closeness, eigenvector centrality measures, determine k-nearest neighbors, and classify users into clusters in both networks and review their differences (Section 3.3, 3.5).
- I demonstrate that the same analysis can be applied to Flickr, online photo-sharing network (Section 3.6)
- I provide a detailed model of energy cost savings by minimizing the number of replicas required for the backup. The model is evaluated with the private email archive, Enron email corpus, and Flickr dataset (Section 3.7).
- I provide limitations of the methodology and analysis (Section 3.8).

3.1.3 Related work

3.1.3.1 One-mode projection of bipartite graph

Laclavík and Šeleng [91, 92] construct a multipartite graph from the email. Each email has its own node with connection to entities extracted from the email such as people, email addresses, phone numbers, dates, etc. Unique entities appear only once in the graph but are connected to each email where they appear. Edges are links between entities representing co-occurrence in the same email part, paragraph, sentence or a composite named entity. Attachments within the email are converted, where possible, to a textual representation. This excludes multimedia attachments like image, video, or audio, which could represent most attachments in private emails.

Guillaume and Latape [69] suggest that interactions of users and information in real world complex networks like Internet, Web, movie actors, co-authors, word's co-occurrence, and protein could be modeled with a bipartite graph. In this graph, users and information are represented by two disjoint vertex sets and edges represent a relationship between users and information.

Wang et al. [150] use one-mode projection of a bipartite graph to capture the similarity of information and shared interests of users. They apply this methodology to an on-line news aggregation site to analyze user's similarity in voting on news stories.

Gupte and Eliassi-Rad in [71] model people and events as nodes and use a bipartite graph where edges represent membership in the event. An event is identified by a set of people that attend it. The focus is on inferring the tie strength using only the graph structure. The problem is to find a function on a bipartite graph that models the tie strength between people, given the bipartite graph representation of people and events.

While bipartite graphs are used in previous research on OSN and email archives, it is novel in the way it is applied in this thesis. It approaches the information interaction between users by connecting users on the basis of sharing an email attachment as the whole rather than a part of it, for instance a url or a name. Naturally, the type of network to use depends on the goal of the research but one clear benefit of sharing the whole attachment is inclusion of images or other binary data, which do not have textual representation.

3.1.3.2 Tie-strength

Tyler et al. [145, p7,p8] set the bidirectional email threshold to 30 with five reciprocal emails and also exclude emails sent to more than 10 recipients.

Kaye et al. [85, p289,p290] set the email threshold to five reciprocal emails. The tie strength corresponds to the frequency of the communication for addresses in *To* and inverse square root for addresses in *Cc* header fields.

Diehl et al. [49, p4] set the email threshold to five reciprocal emails.

Choudhury et al. [41] analyze a family of networks parameterized by a condition on the frequency of the email communication. The edge weight is set to $\sqrt{w_{ij} * w_{ji}}$, where w_{ji} is the number of emails sent per year from i to j. They show that the choice of the threshold significantly impacts the extracted network and formulate the relevance of these networks in terms of the accuracy of prediction tasks dependent on the various network features. They demonstrate that prediction accuracy is maximized over 5-10 reciprocal emails threshold, that for any prediction task the optimal threshold value boost the accuracy up to 30%, and that the optimal threshold is consistent across different datasets and prediction tasks.

Duczynski et al. [103, p3] use cosine similarity as the edge weight. They also filter edges by keeping n most similar neighbors.

Kazienko et al. [86] and Michalski et al. in [108] calculate the tie strength w_{ij} as the fraction of emails sent from node i to node j of all emails sent by node i.

Gilbert and Karahalios in [65] analyze the Facebook friendship relation of 35 participants. Authors identify 74 predictive variables mapped to seven dimensions of tie strength: intensity, intimacy, duration, reciprocal services, structural, emotional support, and social distance. Based on participants answers to tie-strength related questions, authors select the top 15 predictive variables. Analysis shows that intimacy contributes most to the tie strength - 32.8%, followed by intensity 19.7%, duration 16.5%, social distance 13.8%, services 7.9%, emotional support 4.8%, and structural 4.5%. The high ranking of intimacy dimension is corroborated by the previous research of Marsden and Campbell in [99].

Gupte and Eliassi-Rad in [71] infer the tie strength given user participation in a mutual event; i.e. an implicit social network. The event, for instance, can represent a set of people who are recipients of an email. Authors attempt to infer the weighted social network that gives rise to the set of observed events. That is, given the bipartite graph with people as one set of vertices and events as the other set, they want to infer the tie strength between the set of people. The authors present a set of axioms that satisfy any measure of tie strength between two people and theorems describing properties of tie strength functions. They classify tie strength measures found in previous research according to the axioms that they satisfy.

Application of the previous research on the tie strength to the shared attachments network is not obvious. The main issue is that the content of an attachment is not available due to privacy concerns. Consider the top two dimensions discussed by Gilbert and Karaholios in [65]. First is intimacy or exclusivity. Suppose *user1* shares an attachment A with *user2* and shares an attachment B with *user3*, where A is an image from some private event and B is some commonly used emoticon. Clearly A is more exclusive than B, but how do we measure it? Second is the intensity or frequency. If there is one attachment per email message then the frequency is one. But what if there are multiple attachments in the email message then should the frequency be higher? Does it matter if attachments relate to the same event, for instance one party, or multiple events, for instance two different meetings? In order to answer these questions we need to explore a way of inferring the email attachment's properties without violating user privacy.

3.1.3.3 Structure inference

Tyler, et al. [145] define a community structure as a subset of vertices in a email communication network with many edges connecting vertices in the subset and fewer edges connecting subsets. Communities are inferred by repeatedly detecting and removing inter-community edges with large betweenness centrality until the large component is resolved into separate communities. Removing an edge affects the betweenness of other edges. The algorithm removes edges meeting the inter-community criteria at random. The resulted communities are aggregated and analyzed.

Kaye et al. [85] calculate eigenvector centrality, row-sums of a topological overlap matrix, closeness, betweenness, and Opsahl centrality measures for each employee in the Enron core email communication network. The ranked lists of employees are analyzed for consistency with the organizational charts. The results are also visualized with the D3 Java Script library.

Agarwal et al. [17] predict dominance relationships in Enron corpus based on the degree centrality. The algorithm is simple - an employee with a higher degree dominates an employee with a lower degree. Their prediction accuracy is 83%.

Rowe et al. [126] hypothesize that higher communication frequency, shorter email response time, membership in highly ranked cliques, and high centrality measures score identify influential people in the network. They derive a social score calculated as the weighted combination of the metrics and infer the hierarchy, groups of people, and different levels of social hierarchy. The Enron North American West Power Traders extracted network is used for evaluation. The model accurately predicts top level hierarchy and is able to pick two or three most important individuals at any hierarchical level.

Pathak et al. [116] use *tf-idf*[7] score of characteristic words to rank social relations based on their level of concealment. The model correctly identifies concealed relations in Enron email corpus. One interesting finding is that during a crisis period, the communication in general is less concealed than in other periods.

Shetty and Adibi [134] use the Entropy model to identify the most important users in a network. Their hypothesis is that important nodes have the highest effect on graph entropy when they are removed from the graph. The algorithm calculates graph entropy by removing nodes one by one, then ranks the nodes based on their affect on graph entropy. The model is applied to Enron email corpus and the five most influential people are predicted, which correlates with high level management in the organizational chart.

Borgatti [30] shows that existing key measures do not optimally solve the problem of the key people in a network. He identifies two-key player problems. First is the extent to which a network depends on the key player to maintain its cohesiveness, referred to as "Key Player Problem/Negative" (KPP-Neg) and the second is the extent to which the key players are connected to and embedded in the network around them. This problem is referred as "Key Player Problem/Positive" (KPP-Pos). Two metrics are introduced to address each problem:

$$KPP - Neg = 1 - \frac{2\sum_{i>j}\frac{1}{d_{ij}}}{n(n-1)} \text{ and } KPP - Pos = \frac{\sum_j \frac{1}{d_{Kj}}}{n}$$

where d_{ij} is the minimal distance between a pair of nodes ij and d_{Kj} is the minimum distance from any member of set K to node j. The proof of concept is empirically tested on a terrorist and advice-seeking datasets.

Zhou et al. [160] use two metrics to infer pairwise leadership. First is based on the degree disparity concept from Jensen and Neville [82, p10]. The idea is that in a pairwise relationship the leader has a higher proportion of received emails. Second is the conditional probability of a user u_i appearing in a group list given that user u_j is present in the list. The hypothesis is that users who are members on many lists are more prominent and, in a pairwise relation, if X leads over Y then $P(X|Y)$ is high while $P(Y|X)$ is low. The organizational structure is then derived from the pairwise leader relation with a greedy algorithm: by adding a directed edge indicating the leadership relation for the maximal pair $(P(X|Y) - P(Y|X))$ of the selected metric. The algorithm proceeds until there is no users left in the set.

Diesner and Carley [50] time-slice the data to enable longitudinal analysis of the Enron email corpus. They derive top-five ranked lists for closeness, betweenness, eigenvector, and in and out degree centralities in two time slices, one of which is the crisis period. They also derive a list of email exchanged for the two time slices grouped by an employee's position. The main research goal is to analyze network characteristics during crisis situations.

Diehl et al. [49] calculate traffic and content-based mean reciprocal ranking to infer a social relationship. For content-based ranking, they define a master term list in the archive. Then they count the frequency of the term across each communication. They found that content-based ranking consistently outperforms traffic based ranking.

Gupte et al. [72] assume that the link direction in the social network is important in inferring a social hierarchy. The main premise is that a follower is positioned lower in the social hierarchy than a recommender. Authors define a measure to indicate how close the

[7] `https://en.wikipedia.org/wiki/Tf-idf`

given graph is to a true social hierarchy and evaluate the measure in real on-line social networks and random graphs.

Duczynski et al. [103] use degree, closeness, and betweenness centrality measures to infer the organizational hierarchy in the Enron dataset. Their recall value is 14% based on the Enron Gold Standard [17] and they conclude that the hierarchy can not be reliably reconstructed from the communication network.

Namata et al. [111] use communication frequency to classify Enron employees by their organizational titles and broad titles with nine standard classification algorithms: Naive Bayes, Logistic Regression, 1-Nearest Neighbor, 10-Nearest Neighbor, C4.5 Decision Tree, Ripper, ANN, SVM, and Bagging with C4.5. They find that undirected statistics and highest and lowest ranked titles have the best performance. C4.5 class of algorithms performs the best.

Wood et al. [80] consider page-rank, degree centrality, and rooted page-rank together with time dimension to infer manager-subordinate relationship in Enron dataset and advisor-advisee relationship in paper co-authorship. Authors classify actors into Opinion Leaders and Ordinary Users. They demonstrate that time-based methods perform considerably better than ranking-based methods.

Kazienko et al. [86] analyze a mid-size company social network as input to improving managerial processes. The authors conclude that indegree centrality calculated as the sum of normalized edge weight to a node shows good results in predicting department managers.

Michalski et al. [108] extend the work of Kazienko et al. [86] to match an organizational structure to an extracted network. They calculate in and out degree centrality, closeness, betweenness, clustering coefficient, and eigenvector on a mid-size company and Enron social networks. Results show that indegree and eigenvector centrality metrics perform best in predicting the manager-subordinate relationship with 94% and 92% match rate, respectively.

Tang et al. [139] hypothesize that static graphs do not capture temporal characteristics and overestimate the number of connected node pairs and underestimate the path lengths. The authors suggest new metrics for shortest temporal path length, temporal global efficiency, characteristics local temporal clustering coefficient, and temporal local efficiency to capture node interactions over time. The metrics are evaluated on three real world datasets. Results show that the metrics provide a better understanding of the network with respect to the temporal dimension.

Tang et al. [140] propose novel temporal centrality measures of closeness and betweenness that take into account dynamic interactions in a social network over time. Temporal closeness quantifies the speed of information dissemination by a node. Temporal betweenness identifies nodes that act as the key mediators between the most communication paths over time. The metrics uncover important nodes that are better for information spreading and nodes playing vital role in mediating between the most communication channels.

Wu and Chen in [153] classify users according to their actions, like click, post, share, show, etc. in 17salsa.net OSN. The normalized feature vector is chosen based on the subset of user's actions. Users then are clustered into related groups with k-means algorithm. The number of stable clusters is chosen empirically from the set of {2, 3, 4, 5, 6}.

Zhang et al. in [157] analyze the behavior of email users related to their participation in various email subjects. They extract multidimensional network from Enron email corpus. The vertices in the network represent users, email subjects, and keywords in email subjects. Edges represent the relation between users and subjects and between subjects and keywords. Jaccard distance is calculated to infer keywords and subject similarity. Other features are defined: active user behavior as the probability that the user sends a specific subject, passive user behavior as the probability that the user receives a specific subject, subject influence ability as the ratio of certain types of users sending one type of subject to all users sending this type of subject, and subject propagation ability as the ratio of certain types of users receiving one type of subject to all users receiving this type of subject. A keyword selection algorithm based on the greedy approach is used to select the most meaningful keywords. Finally the k-means algorithm is used to obtain keyword clusters and derive statistics for a user's participation, grouped by their titles, in email subjects.

Yelupula and Ramaswamy in [155] derive features from the Enron email corpus in order to rank employees into different organizational echelons and divide into clusters with k-means clustering. Features calculated over the sending window are the number of emails sent, number of distinct subjects sent by a sender, number of unique email recipients, and number of unique sender addresses. Features calculated over the receiving window are the number of emails received, and number of emails received in *To, Cc, Bcc*. Further, normalized weight is calculated for sent and received (as sum of *To, Cc, Bcc*) emails for each employee and for clusters of employees. The normalized weight W of the entire dataset is calculated and employees are classified based on their weight into three echelons defined by ranges $\{0\text{-}W, W\text{-}2W, 2W\text{-}3W\}$. Empirically authors find that features such as weight, number of emails received in *To, Cc, Bcc*, and number of different subjects sent by each member produce the best groups with k-means clustering. Achieved accuracy for four broad clusters of first level active management, active employees, less active management, and miscellaneous group is 80%, 75%, 95.65%, and 72.22%, respectively.

Maia et al. in [98] use k-means clustering to group YouTube users that share a similar behavior pattern. In the crawled dataset, users are linked if they subscribe to videos of other users. In the extracted network, a directed edge from user A to user B means that A subscribed to videos of B. This feature is a unidimensional vector of length nine composed of individual and social information pertained to the user: number of uploads, number of watches, number of channel views, join date, age, clustering coefficient, reciprocity, outdegree, and indegree. The data is normalized by the maximum value of each feature. K-means with Euclidean distance measure is then used to cluster users with similar behavior. Clustering is done incrementally with an increasing number of clusters. Clusters are merged if the difference between centroids exceeds a defined threshold. The algorithm stops when a newly created cluster merges with an already merged one. The clustering identifies five groups of similar behavior.

As shown above, there are different types of structure inference methodology that could be used in SNA depending on the goals of the research. Metrics such as degree, betweenness, closeness, and eigenvector centrality measures are commonly used as standalone to make predictions about the most structurally influential nodes in the network or as features in classification methods [145, 85, 17, 50, 103, 80, 86, 108, 140]. I use all of the above centrality measures in this thesis. K-means is a common algorithm for clustering users into groups based on some similarity features [153, 157, 155, 98]. K-nearest neighbor classification is also used in SNA research [111]. Both methodologies are some of the simplest in ML and well suited to identifying groups of people with similar properties, which in this research is those attachments shared between users.

3.1.3.4 SNA application for content placement and routing

Sastry and Crowcroft in SpinThrift [128] arrange the data so as to skew the majority of access to fewer disks, allowing other disks to be spun down, and consequently conserving energy. SpinThrift analyzes information from the social network to identify viral and non-viral access to distinguish popular and non-popular items. The evaluation on Digg and Vimeo OSN showes that SpinThrift achieves the same power consumption comparable to a scheme that computes an optimal ordering of items based on strict popularity ranking and reduces the number of items that need to be re-arranged. SpinThrift relies on the friend's list, not available in email, to infer viral and non-viral item access. In addition, popularity is not applicable to email. Moreover, re-arranging the content for access from fewer disks does not map to the IoT home environment with fewer devices.

Sastry et al. in Buzztraq [129] use social cascade prediction to generate hints for user generated content placement closer to future access. The goal is to mitigate the difficulty of placing content not popular enough to be globally replicated. Traditionally, in location based replication, the content is replicated to geographical regions which contributed the maximum number of users in the past. Buzztraq considers the social aspect of access to the content's by expecting that future access will be made by friends of previous users. It maps a user's affiliation to geographical locations, which are clustered into regions using geodesic distances. Evaluation of Buzztraq on Facebook user's profiles shows that initially when there is no detectable social cascade, location-based placement is better. However, as the number of accesses increases, social cascade prediction outperforms location prediction. In the case of decentralized email architecture, locality of access is achieved by storing email messages on the user's IoT devices. But the placement of additional replicas should be based on with whom the user has the most shareable units of data (e.g., email attachments) so as to optimize backup cost, rather than on the geodesic proximity. The latter can be used as a secondary parameter in the optimization decision.

Hui et al. in BUBBLE Rap [78] propose a social-based forwarding algorithm. The algorithm selects high centrality nodes and community members of destination as relays. A source node bubbles the message through the global ranking tree using global ranking until the message reaches a node in the same community as the destination node. Then the message bubbles through the local ranking tree until it reaches the destination node or the message expires. To reduce the cost, once the message is delivered to the community, the original carrier deletes the message. The algorithm shows significant improvement in forwarding efficiency. In a decentralized email architecture I assume that there is a direct connection between sender and receiver. BUBBLE Rap can improve availability when the direct connection link is down by routing the message via trusted friends, especially in the case of an email list. This requires further research to make the routing efficient in terms of energy and latency. In the former, replicating the message between devices is energy consuming especially if the message contains large attachments. In the latter, the latency may exceed a user's tolerance to a delay in the message delivery.

Thilakarathna et al. in [143] propose a friend-to-friend content dissemination system. The system takes advantage of trusted social networking friends. First, communities are detected with the k-clique algorithm. Second, the consumer with the highest dynamic centrality metric is selected as the helper responsible for content propagation within the community. Then the next consumer with the highest centrality is selected in the next community. The process repeats until all communities have helpers assigned. It is expected that the helpers become quasi-static over time due to the regular behavior of people. Consequently, helpers need to be selected only for newly added users. Evaluation shows that the algorithm results in a delivery success rate of up to 80% with less than 10% replication. But in a social networks the published content is generally available to all friends and email has a more private characteristic as shown in Section 3.2 with fewer "consumers". The algorithm can increase email availability but with higher energy cost by replicating to the "helper" in each community and higher latency (three-day delivery deadline). Moreover, it does not take into consideration the size of the message, which affects the bandwidth and the storage.

3.1.4 Problem formulation

The question that I ask in this chapter is whether SNA of the network extracted from an email archive can be used for optimization of the decentralized email architecture. I am proposing to extract the network from email attachments shared between users. This is a novel approach in the context of the email archive where conventionally the network is extracted either from email communication or from embedded data, for instance names, phone numbers, or url. The motivation for this approach is that, based on statistics of two email archives used in this dissertation, the majority of email data is represented by email attachments. Consequently, optimization of the attachment storage may reduce disk usage, energy, and bandwidth. Section 3.2 and 3.4 propose rules for network extraction based on data analysis of related properties. Sections 3.3, 3.5 analyze network and node level metrics for conventional communication and shared attachments networks. Section 3.7 shows application of the analysis to optimization of the decentralized email architecture. I also demonstrate in Section 3.6 how the same kind of analysis can be applied towards the photo-sharing Flickr social network.

3.2 Network extraction

I construct the graph used to analyze a social network in two ways. First is via communication between users, where nodes represent the email address or the user listed in *From, To, Cc*, and *Bcc* email header fields. Edges represent the relationship between the sender (*From*) and recipients (*To, Cc, Bcc*). Edges are undirected and weighted by the frequency of communication. Second is via shared attachments by constructing a one-mode projection graph on users of a bipartite graph as demonstrated in Figure 3.1. The top figure shows the bipartite graph extracted from an email sent from *user-from* to two users *user1-to* and *user2-to*. The email contains two attachments *attachment1* and *attachment2*. The bottom figure shows the corresponding one-mode projection graph on users, where nodes represent users. Edges represent attachment sharing between users. The edges are undirected and weighted by the number of shared attachments. The edges are undirected because the direction is not meaningful in all cases. Consider a user U sending an email with an attachment A to users $U1$ and $U2$. In the extracted one mode projection graph on users, the edges $\{U,U1\}$, $\{U,U2\}$ can have a direction but the edge $\{U1,U2\}$ can not. Moreover, it is possible that a user $U1$ sends an email with an attachment A to a user $U2$ and another user $U3$ independently sends an email with the same attachment A to a user $U4$. In this case edges $\{U1,U2\}$, $\{U3,U4\}$ can have a direction and edges $\{U1,U4\}$, $\{U2,U4\}$, $\{U1,U3\}$, $\{U3,U2\}$ can not have a direction.

To extract the graph from an archive, for each user U's archive A, I extract email

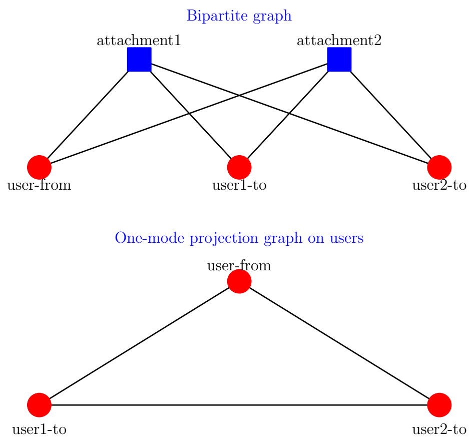


Figure 3.1: Bipartite graph with attachments and users vertices.

messages which contain attachments. For each attachment in the extracted email, a global dictionary is maintained keyed by 1) the attachment; 2) user U and every user V in the email's *From, To, Cc*, and *Bcc* header fields distinct from the user U; 3) the email's *Message-ID*. For each attachment in the global dictionary I look at all unique user pairs and create nodes for these users, if nodes do not already exist, and the edge connecting these nodes. By aggregating sender and recipient users in the same pool I not only capturing direct communication between users, for instance a user in *From* sends email to a user in *To*, but also capture a friend-of-a-friend (FOAF) relationship; i.e., all users in *To, Cc*, and *Bcc* who may not communicate directly but become connected via shared attachments. In addition, by capturing a user U, who owns archive A, we may capture relationships between user U to users in *From, To, Cc*, and *Bcc* when none of these users is the user U.

An important part of extracting the Social Network is defining the strength of ties or the weight of edges. As noted in Choudhury et al. [41, p2], tie strength itself remains an ambiguous concept with multiple, possibly inconsistent, definitions and that there is a non-trivial range of thresholds of 5-10 reciprocated emails per year which maximizes prediction of the relevant task that depends on various network features. Some researchers simply count the frequency of communication as, for example, Agarwal et al. [17, p3]. Others consider *Cc* communication less important than *To* and decrease it at an inverse square-root rate as in Kaye et al. [85, p6]. And yet others use an email frequency threshold, which is typically set to 5 emails as in Shetty and Adibi [135, p7] but could be even higher like 30 in Tyler et al. [145, p7]. How should tie strength be defined when the network is extracted from shared attachments? The size of the attachment should not contribute to the tie strength. Indeed, an event is shared via the attachment regardless of the image resolution or the document length. Consequently, I suggest assigning a weight of 1 to the attachment. But should the weight be aggregated over all attachments in the email or should it be 1 regardless of the number of attachments? Either approach seems sensible. Suppose Alice sends 10 New Year Party pictures to Bob. Alice and Bob share one event so the weight should be 1. But if Alice sends 10 pictures from 10 different events in the same email, then the weight should be 10. I am going to make the assumption that an email generally contains attachments related to one event and one shared attachment creates one tie between two users. All other attachments from the same unique email are ignored. But if an attachment from the email appears in another unique email then this attachment will be considered as another tie.

Another issue to consider when extracting the network from shared attachments is the value of the attachment to the analysis. For instance, with the present day proliferation of e-commerce and growth of machine-generated emails, it is possible that many users have the same e-site logos in their email messages, like Amazon logo. This does not make all users of Amazon socially connected. Overall, this kind of attachment could be a company logo, an e-signature, a common document like a benefit form, and an Internet trend or rumour where a trendy multimedia or a document are spread to many users in the network. I jointly call these attachments TRAM (TRend+spAM). How can we filter out TRAM attachments?

I analyzed the Enron email corpus and private email archives that used in my research for the pattern of attachments spread in the network. To do this, for each shared attachment I collected unique senders (users in *From*) and for each sender I collected recipients (users in *To,Cc,Bcc*). Then, for each attachment, the spread is inferred in the following way:

- From the list of senders I select the ones that do not appear on any of the recipient's list. Those senders are considered the source of the attachment and become the root node of each spread tree. My assumption is that the reply email does not contain the attachment.
- For each sender in the root's list, I loop over the recipient's list and add them as a child node.
- If the recipient appears in the sender's list, it becomes the parent node. I iterate over the recipient's list of this node and add them as a child node.
- The step above repeats until all senders are visited. To avoid cycles, I visit each sender only once. I also visit each recipient only once to exclude the case of the recipient replying back to the email list.

The analysis includes all users having shared attachments, not just the core users.

The inferred patterns can be categorized into four groups as displayed in Figure 3.2. Group (a) and (c) have a single root and (b) and (d) have multiple roots. Multiple roots is the case when the same email attachment is sent independently by multiple unique senders. Groups (c) and (d) can have depth greater or equal to two. Table 3.2 shows the percentage of the root pattern cases. There are up to 42 and 26 multiple roots in Enron and private archives, respectively. I reviewed 10 attachments with roots higher than 30 from the Enron archive. All of them were common url's with the size of the attachment less than 100 bytes. I reviewed seven attachments with multiple roots higher than 10 from the private archives. I was able to review those attachments because they were in my email's archive. All of those attachments were either a logo image or emoji. Three attachments out of seven were less than 1 KB and two were less than 10 KB. Overall, multiple root cases are relatively few. As we can see, the single root in both Enron and private archives accounts for at least 94.43% of the spread pattern. Both single and multiple root cases can uncover hidden relationships between users who do not communicate with each other. Consider these use cases:

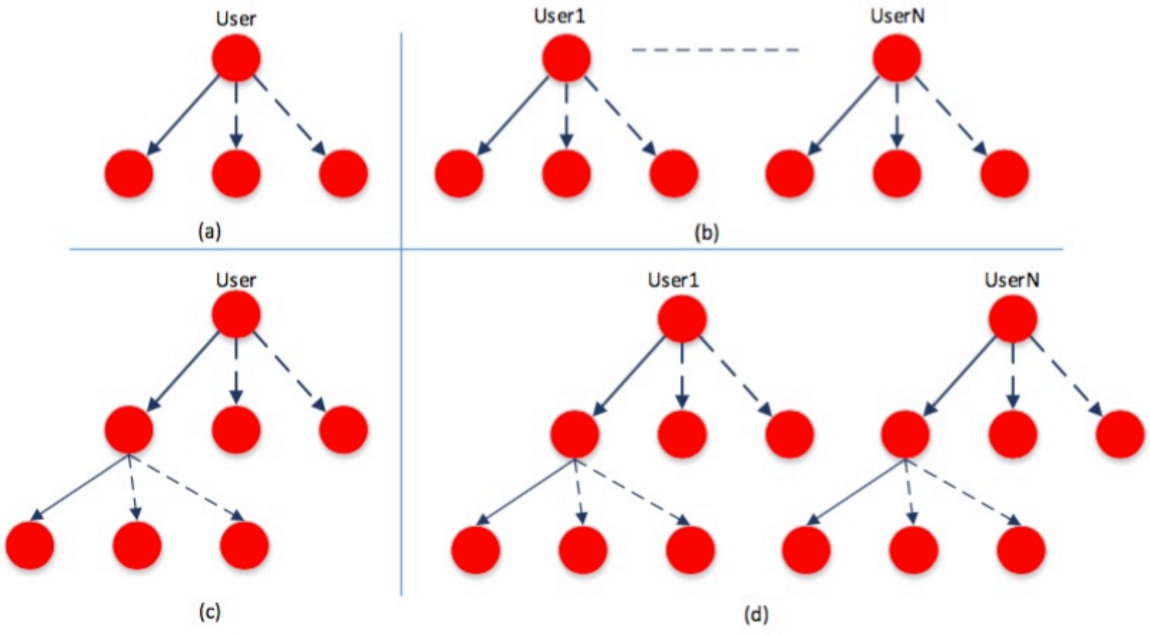


Figure 3.2: Attachment spread pattern. (a) Single root, depth 1; (b) Multiple roots, depth 1; (c) Single root, depth >= 2; (d) Multiple roots, depth >= 2. Multiple roots is the case when the same email attachment is sent independently by multiple unique senders.

Table 3.2: Root pattern percentage.

	1	2	>= 3
Enron	97.02	2.43	0.55 (3-42)
Private	94.43	4.22	1.35 (3-26)

- Single root. *User1* sends an email with the same attachment to *User2* and *User3*. *User2* does not communicate with *User3*. We can infer a possible relationship between *User2* and *User3* because they share the same email attachment.
- Multiple roots. *User1* sends an email with attachment to *User2* and *User3* sends an email with the same attachment to *User4*. We can infer a possible relationship between *User1, User2, User3*, and *User4* because they share the same email attachment.

The depth of the spread is displayed in Table 3.3. What we see is that in at least 91.67% of cases an attachment reaches a recipient in one hop. The cases of hops greater than one are either caused by forwarding of an attachment to other recipients of replying to the same or modified list. These cases may establish new ties of type FOAF users who do not communicate directly.

Table 3.4 shows the top 10 patterns of spread. The meaning of colon-separated values

Table 3.3: Depths pattern percentage.

	1	2	3	>= 4
Enron	94.19	5.43	0.34	0.04 (4-6)
Private	91.67	6.88	1.08	0.37 (4-11)

Table 3.4: Percentage of pattern spread.

Enron		Private archive	
Pattern	%	Pattern	%
other	6.07	other	5.07
1:80	0.45	3:20	0.54
1:50	0.64	1:10:10:10	0.55
3:10	0.72	1:20:10	0.59
1:200	0.75	1:30	0.85
1:40	1.14	2:20	1.09
1:30	2.40	3:10	1.88
2:10	3.70	1:10:10	4.70
1:10:10	3.92	2:10	5.80
1:20	5.41	1:20	5.86
1:10	74.79	1:10	73.07

Table 3.5: Attachments distribution in unique messages.

	1	2	3	4	5	>= 6
Enron	76.94	14.48	3.83	1.81	0.96	1.98
Private	76.74	13.62	4.63	2.00	1.08	1.93

Xr:X1(:X2)+ in the column labeled Pattern is: *Xr* is the number of roots, *X1* is the number of recipients at depth one, *X2* is the number of recipients at depth two, and so on. I aggregated recipients into bins of length 10 with the value indicating the upper boundary of the bin. For instance, 1:10:10 means the pattern of attachment spread with the number of roots equal to one, number of recipients at depth one is 1-10, and number of recipients at depth two is 1-10. As we see, at least 73.07% of cases fall into the pattern of one user sending an attachment to up to 10 recipients. It is interesting that both Enron and private archives have the same top four pattern spread. Table 3.4 shows only the top 10 patterns. The rest of the patterns account for 6.07% and 5.07% for Enron and private archives, respectively. From Table 3.3 we can tell that the pattern can go up to 6 and 11 depths for Enron and private archives, respectively.

In addition to the pattern spread analysis, I looked at the distribution of attachments in unique messages and the distribution of email lists. Table 3.5 shows percentage of attachments distribution in unique messages. We see that at least 76.74% of attachments are sent in one message.

Table 3.6 shows distribution of the email lists. Up to a value of 100, the lists are aggregated into bins of 10, and after that the value of 100 in bins of 100. Both distributions appear to be similar in not only having the most email sent to under 10 recipients but also having ranges of list size which are close, including the highest 900. This is in spite of the substantial difference in number of users in each dataset. The number of core users in the private archive is almost five times less than the Enron's number of core-users. It is interesting that both Enron and private archives have the same top four email lists.

The conclusion from the analysis above is that spread of attachments in email is very targeted and stresses the private nature of email communication. Indeed, we see that about 94% of attachments are sent from a single sender; 91% of attachments are not forwarded by their recipients; 73% of attachments are sent to a group of up to 10 people

Table 3.6: Distribution of email lists.

Enron		Private archive	
List size	%	List size	%
700	0.00	-	-
900	0.01	400	0.00
400	0.02	600	0.00
500	0.02	500	0.01
300	0.29	900	0.01
90	0.37	60	0.07
70	0.39	70	0.07
100	0.41	90	0.07
60	0.41	100	0.10
80	0.43	200	0.11
50	0.87	80	0.13
200	1.56	50	0.20
40	1.65	40	0.33
30	2.76	30	0.73
20	8.88	20	4.22
10	81.95	10	93.97

without further forwarding; 76% of attachments are sent in a single unique message. Finally, while there are email lists with a large number of up to 900 users, 82% of emails sharing attachments have an email list up to 10 users. An interesting fact is that in the private archives this number is 94%, even though again there are very large email lists of up to 900 users. This might be explained by a smaller group of core users in the private archive. But it could also mean that personal email is more private. Another interesting fact is that private email communication appears to be more gossipy. This can be seen from the higher number of multiple roots (Table 3.2), higher number of depths (Table 3.3), and higher number of forwarded emails with attachments, which in the private archive is 8.33%, while in Enron it is 5.81% (inferred from the pattern spread with depth greater than one). We also see that the pattern of attachment spread, with the exception of the email lists, is remarkably similar. We could make an assumption that the extracted attachments network based on the low boundary threshold; i.e., following the highest distribution per above data, will have network characteristics similar to the communication network. Indeed, in this case we will generally establish similar ties as in the communication network. Consequently, we can use the characteristics of the communication network to tune extraction of the attachments network or, at a minimum, establish some low-boundary thresholds.

Based on the above analysis, I suggest the following approach to filtering out non-useful attachments:

- A threshold on the attachment's size may filter out common logos and e-signatures. Attachments of those type generally have a small size. Figure 3.3 shows Enron's attachment size histogram. There are two spikes at 0.1 KB and 0.5 KB. I reviewed a Simple Random Sample (SRS) of attachments with the size less than 1 KB. I found that all attachments except for one were TRAM. Most of these attachments are artifacts of the EDRM building of the Enron dataset with attachments, not

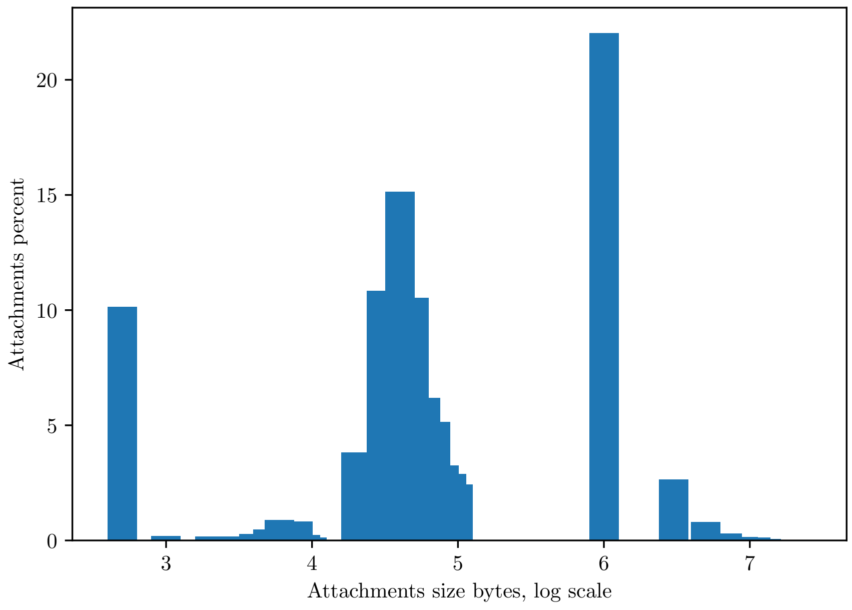


Figure 3.3: Attachments size histogram for Enron corpus.

the linking processing. Those attachments are either documents with text stating that the attachment's link was not found or documents that cannot be opened. Others are e-business cards, logos, executable files, news and e-site urls. I ignore an attachment if its size is less or equal than 1 KB. The extracted network has average degree equal to 59.6 and clustering 0.714 as opposed to the unfiltered network with average degree 61.12 and clustering 0.716. Removing small size attachments has a small effect on the network connectivity. Figure 3.4 shows the attachment size histogram in the private archive. It does not have an apparent spike like in Figure 3.3. In the private archive, data under 10 KB are filtered, which appears to be "noise" and accounts for 6.35% of the attachments. For privacy reasons the SRS of the data cannot be extracted or analyzed if the attachments fall into the TRAM category. The resulting extracted network has average degree 12.759 and clustering 0.678 as opposed to the unfiltered network with average degree 16.55 and clustering 0.767.

- Bulk email can significantly affect network's connectivity and clustering. Indeed, sending a broadcast message with an attachment of the company's quarterly earnings will make all employees share the same attachment; but it will not make all of them socially connected. Figure 3.5 shows average degree and clustering depending on the filtered bulk email list size in the Enron's corpus. There are two interesting points with the list sizes of 200 and 900. They are explained by four emails sent to 193 users and two and three emails sent to 947 and 948 users respectively. Clearly 900 can be classified as the bulk email group. But it is not obvious what the low threshold should be. Dunbar in [55, p10] suggests social group sizes of 5, 12, 35, 150, 500, and 2,000 with 150 being a cognitive limit also known as Dunbar's number. Hill and Dunbar in [77, p15] further categorize the groups as support cliques, sympathy groups, bands, and higher-level groupings (above 35). In [54, p6], Dunbar defines

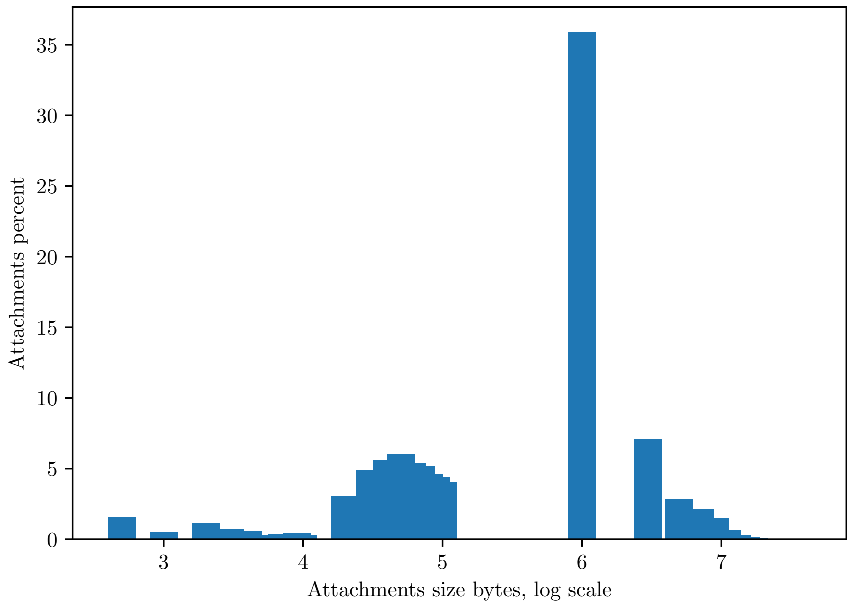


Figure 3.4: Attachments size histogram for private archive.

band group size as 30-50 individuals. Dunbar then suggests in [56, p1] that there is a top 50 (corresponding to the band group) with whom we keep up every month or so; all others are those with whom we correspond in any meaningful way. The 200 email-size falls into higher level grouping and can be categorized as bulk email. I am therefore suggesting to set the band group size of 35 as the bulk email threshold in the Enron corpus. This threshold corresponds to the extracted network with average degree 29.4 and clustering 0.566. This is a change of 52% and 21% for average degree and clustering respectively. Figure 3.6 shows average degree and clustering depending on the filtered bulk email list size in private archive. Both curves are smoother as compared to the Enron ones in Figure 3.5. The bands threshold of 35 can not be applied to the private email archives with the core size of 29. Consequently, I am applying the sympathy group threshold of 12 to the private archive. The resulting extracted network has average degree 14.069 and clustering 0.698.

- Other properties that influence network's connectivity and clustering are the frequency of a shared attachment in unique emails and the frequency of unique senders of the shared attachment. Consider the Amazon logo example I mentioned above - many users have the same logo attachment in their unique emails. Another example is an intra-company communication with email containing the company's logo. In this case many unique emails have the same attachment but also the same attachment is sent by many unique senders. A news story or a rumor-trend gone viral with the message being re-sent by multiple users is similar to the company's logo example. Figure 3.7 shows average degree and clustering depending on the filtered attachment and sender frequency in Enron corpus. There is a point at the frequency 3 with a sharp change in the average degree from 35 to 53 and the clustering from 0.67 to 0.72. Number 3 is not coincidental for both the attachment and the sender frequency. In the former, the attachment in three unique messages with three unique

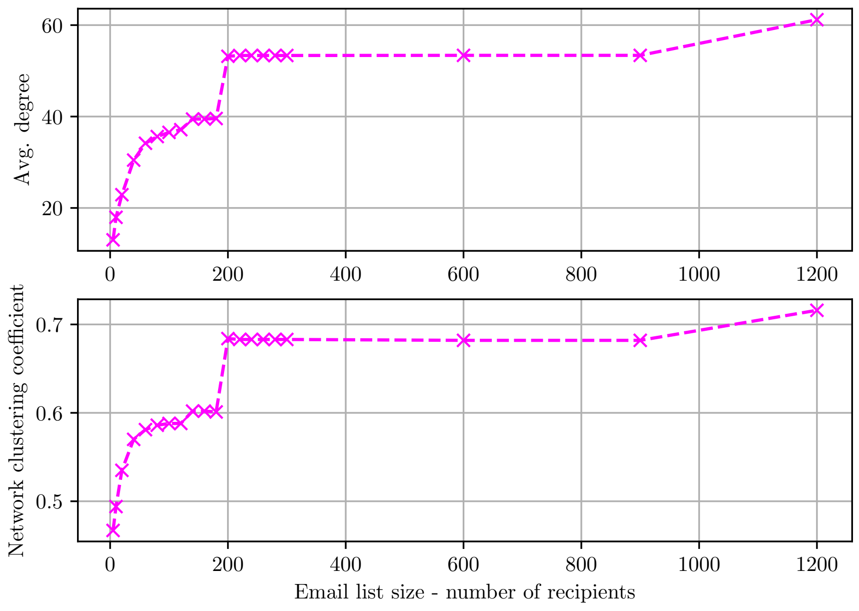


Figure 3.5: Avg. degree and clustering depending on the filtered bulk email in Enron corpus.

senders or receivers creates the closed triad. And in the latter, the attachment with three unique senders always creates a closed triad. Consequently, we see higher clustering and degree when the frequency is greater than 3. An attachment's frequency has higher clustering because the set of attachments with frequency 3 or higher is a superset of senders with frequency 3 or higher. Indeed, the attachment with three unique senders will occur only in unique messages. I filter out attachments with frequencies higher than 2. The network extracted this way has average degree of 37.32 and clustering of 0.671. Figure 3.8 shows average degree and clustering depending on the filtered frequency of the attachment and the sender in the private archives. There is a noticeable point at frequency 2 of the clustering curve. Also, what is interesting is that the clustering of an attachment's frequency increases from the frequency 1 to 2 and then decreases at the frequency 3 and stays relatively flat until frequency 10. Figure 3.9 shows the extracted attachments network for filtered attachment's frequency greater than 1, 2, 3, and 10. At frequency 1 (Figure 3.9a), the extracted network has three components, consisting of two dyads and a larger component with two leaf nodes. At frequency 2 (Figure 3.9b), the number of components is still three with two dyads and a larger component, but the leaf node 20 is now part of the closed triad, and the added edges create more triads overall, consequently the clustering is higher. Both networks (Figure 3.9a) and (Figure 3.9b) have 26 nodes. At frequency 3 (Figure 3.9c) the number of nodes increases to 29. There are three components, consisting of a dyad, an opened triad, and a larger component with two leaf nodes. Also added nodes and edges lower the number of closed triads; consequently, the clustering decreases. Finally, at frequency 10 (Figure 3.9d), there is only one component with four leaf nodes and added edges increase the number of closed triads; consequently, the clustering increases. I apply the same low boundary

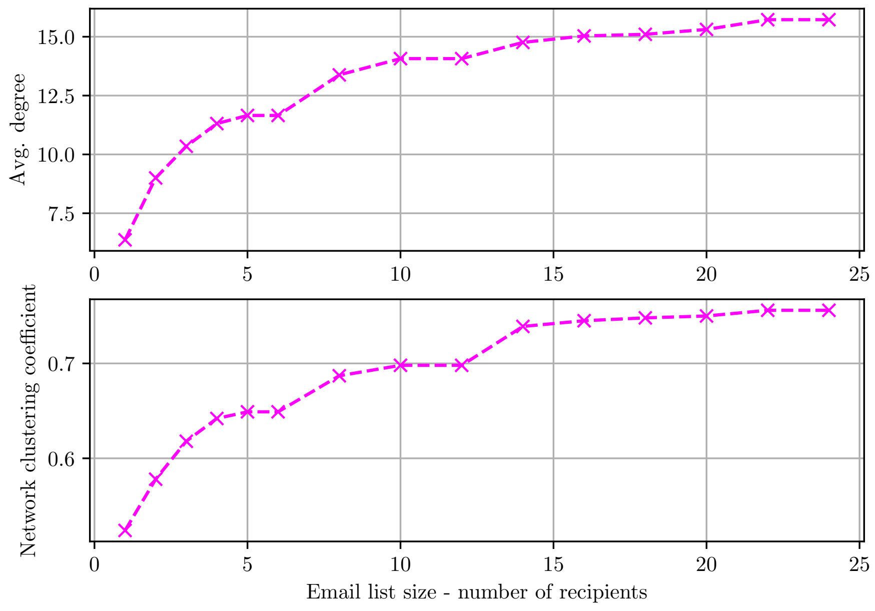


Figure 3.6: Avg. degree and clustering depending on the filtered bulk email in private archive.

of the attachment's frequency of 2 and the attachment's unique sender frequency of 2 to the private archive. The resulting extracted network has the average degree 10.074 and clustering 0.677.

Note that the above approach at best establishes the low boundary of described properties and may eliminate some common logo, signature, or trend attachments but does not ensure the quality of an attachment and also removes some valuable attachments in the process. It generally limits the cases of attachment-sharing to a single email sent to a list of at most 35 users or emails forwarded only once. A better solution might be to use methodology similar to SPAM detection based on Social Networks as in Lan and Yeung [93], unsupervised ML, or information spread.

3.3 Enron network analysis

I calculate graph-level statistics and node level statistics of degree, eigenvector, betweenness, and closeness centrality measures and k-nearest neighbor classification and k-means clustering for both networks. Enron's organizational charts are used throughout the analysis. The charts are a combination of previous research of Agarwal et al. [17], Hardin and Urc [75, p17], documents related to Enron's legal proceedings[8], Enron emails, and information I discovered on LinkedIn[9]. I use Python's NetworkX module[10] for overall network statistics and centrality measures, Graphlab module for k-nearest neighbor classifier[11],

[8]https://www.gpo.gov/fdsys/pkg/GPO-CPRT-JCS-3-03/pdf/GPO-CPRT-JCS-3-03-3-2-8.pdf
[9]https://www.linkedin.com
[10]https://networkx.github.io
[11]https://turi.com/learn/userguide/index.html

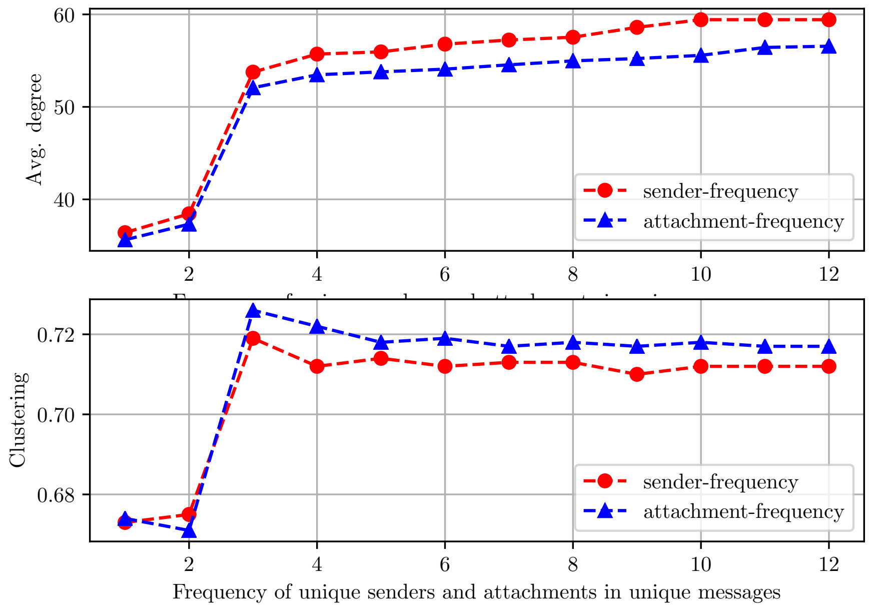


Figure 3.7: Avg. degree and clustering depending on the filtered frequency of the attachment and the sender in Enron corpus.

and Sklearn.cluster[12] for k-means clustering.

3.3.1 General statistics

Table 3.7 shows general statistics for both networks. For the attachments network, statistics with the filtered and un-filtered attachments are shown. We see that the filtered attachments network is less connected, less populated with edges, and less centralized as compared to the communication network. It has one fewer node. This node does not have any shared attachments with other employees when filters are applied. The unfiltered attachments network is significantly different from the other two. Most notably the clustering coefficient, characteristic path length, average number of neighbors, and network density all point towards a network with more triads and more populated with edges. This is the result of discovering more FOAF relationship by connecting users via attachments shared in their email archives. In essence the definition of friend is extended from *someone who has direct communication with me* to *someone who has some information shared with me*. In the extreme case, when bulk email is sent to everyone, everyone shares the same information, resulting in a complete graph.

3.3.2 Centrality measures analysis

Table 3.8 shows the ranking of top 10 employees in at least one of the following: degree (D), eigenvector (EV), betweenness (B), and closeness (C) centrality measures in communication network. The overall rank was calculated as the sum of inverse value of ranking in the centrality measure plus one for each centrality measure which is in the top 10 in

[12]http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Table 3.7: General statistics for communication and attachments networks.

Statistics	Communication network	Attachments network, filtered	Attachments network, unfiltered
Clustering coefficient	0.545	0.566	0.716
Connected components	1	1	1
Network diameter	4	5	3
Network radius	2	3	2
Network centralization	0.448	0.287	0.407
Characteristic path length	2.025	2.295	1.62
Avg. number of neighbors	24.12	20.04	61.12
Number of nodes	150	149	150
Number of edges	36 485	11 408	33 780
Network density	0.162	0.135	0.41
Network heterogeneity	0.588	0.567	0.483

Table 3.8: Top 10 employees for centrality measures in communication network.

Overall Rank	Name	Title	Centrality Measures
1	John Lavorato	COO, EA	D(7),EV(1),B(2),C(1)
2	Liz Taylor	Assistant to President	D(55),EV(2),B(1),C(2)
3	Louise Kitchen	President&CEO, EN	D(9),EV(4),B(6),C(5)
4	Sally Beck	VP ENA EO	D(48),EV(3),B(3),C(3)
5	Kenneth Lay	CEO	D(50),EV(5),B(5),C(4)
6	Jeff Dasovich	Dir Stat Gov Affairs	D(1),EV(23),B(8),C(11)
7	Phillip Allen	Managing Dir Trading	D(33),EV(6),B(15),C(6)
8	Kevin Presto	VP Trading, ENA EP	D(30),EV(7),B(11),C(7)
9	Mike Grigsby	VP Trading, ENA GW	D(11),EV(10),B(16),C(8)
10	Scott Neal	VP Trading, ENA GE	D(38),EV(8),B(20),C(9)
11	David Delainey	CEO ENA&EA	D(17),EV(9),B(28),C(10)
12	Tana Jones	Sr Lgl Spclst, Networks	D(2),EV(26),B(29),C(22)
13	James Steffes	VP Gov Affairs	D(3),EV(21),B(26),C(15)
14	Mark Taylor	VP & GC, Networks	D(5),EV(14),B(13),C(13)
15	Susan Scott	Assistant Trader	D(16),EV(30),B(4),C(14)
16	Sara Shackleton	VP & Sr Cnsl, ENA	D(4),EV(64),B(68),C(79)
17	Richard Shapiro	VP Reg Affairs	D(6),EV(28),B(79),C(29)
18	Steven Kean	VP & Chief of Staff	D(8),EV(19),B(60),C(21)
19	Bill Williams	Trader	D(62),EV(120),B(7),C(88)
20	John Forney	Mngr Real Time Trading	D(81),EV(88),B(9),C(36)
21	Lysa Akin	Sr Adm Asst Gov Affairs	D(73),EV(81),B(10),C(63)
22	Carol Clair	AGC, Networks	D(10),EV(103),B(128),C(119)

Table 3.9: Top 10 employees for centrality measures in attachments network.

Overall Rank	Name	Title	Centrality Measures
1	Phillip Allen(7){1}	Managing Dir Trading	D(12),EV(1),B(2),C(1)
2	James Steffes(13){7}	VP Gov Affairs	D(1),EV(18),B(6),C(3)
3	Mike Grigsby(9){3}	VP Trading, ENA GW	D(14),EV(2),B(4),C(2)
4	Hunter Shively{6}	VP Trading, ENA GC	D(46),EV(3),B(3),C(5)
5	John Lavorato(1){10}	COO, EA	D(19),EV(4),B(8),C(4)
6	Elizabeth Sager{16}	VP & AGC, ENA PT	D(8),EV(33),B(7),C(10)
7	Susan Scott(15)	Assistant Trader	D(11),EV(13),B(1),C(8)
8	Keith Holst{19}	Trader	D(20),EV(5),B(12),C(6)
9	Steven Kean(18){12}	VP & Chief of Staff	D(4),EV(14),B(25),C(9)
10	Tana Jones(12){17}	SR Lgl Spclst, Networks	D(2),EV(71),B(71),C(51)
11	Richard Shapiro(17){14}	VP Reg Affairs	D(3),EV(22),B(36),C(12)
12	Jeff Dasovich(6){13}	Dir State Gov Affairs	D(5),EV(28),B(24),C(17)
13	Kevin Presto{5}	VP Trading, ENA EP	D(44),EV(19),B(11),C(7)
14	Matthew Lenhart{9}	Analyst	D(18),EV(10),B(19),C(11)
15	Barry Tycholiz{8}	VP Trading, ENA GW	D(16),EV(7),B(34),C(19)
16	Williams Jason	Trader, ENA GC	D(39),EV(8),B(15),C(24)
17	Thomas Martin	VP Trading, ENA GT	D(66),EV(6),B(40),C(23)
18	Jay Reitmeyer{11}	Associate	D(23),EV(9),B(43),C(15)
19	Mike Swerzbin	VP Trading, ENA WP	D(117),EV(76),B(5),C(50)
20	Mark Taylor{2}	VP & GC, Networks	D(6),EV(47),B(45),C(33)
21	Marie Heard	Spclst Lgl, ENA	D(7),EV(80),B(85),C(61)
22	Sara Shackleton(16)	VP & Sr Cnsl, Networks	D(9),EV(69),B(48),C(34)
23	Robert Badeer	Mgr Trading, ENA WP	D(67),EV(60),B(9),C(32)
24	John Forney	Dir Trading, ENA EP	D(100),EV(94),B(10),C(38)
25	Stacy Dickson	Sr Cnsl, ENA	D(10),EV(100),B(80),C(94)

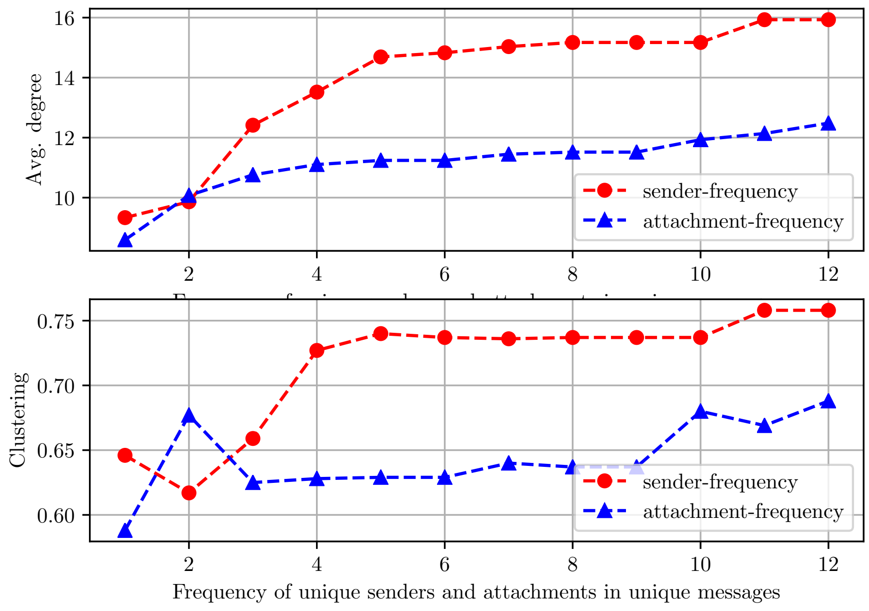
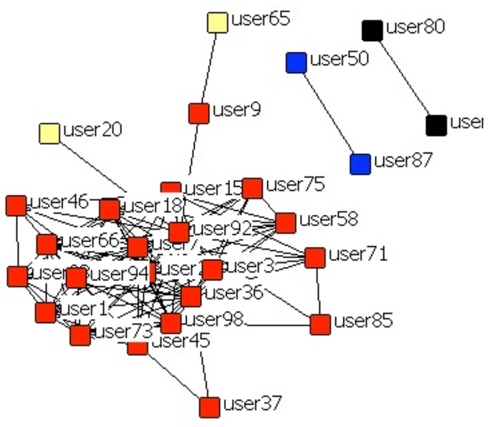


Figure 3.8: Avg. degree and clustering depending on the filtered frequency of the attachment and the sender in private archive.

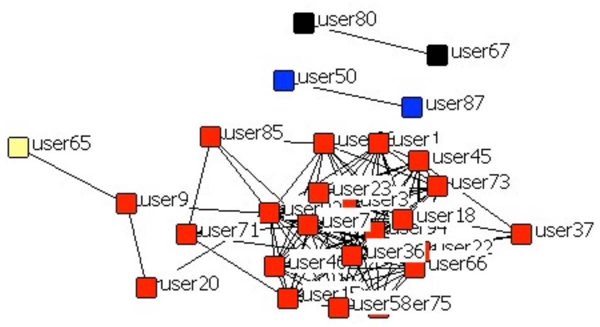
order to favor employees who have more top 10 rankings. It is not surprising that the most influential employees are Executives. Number one is John Lavorato, COO Enron Americas. Liz Taylor, number two, is assistant to President & COO Greg Whalley, who is not ranked in any top 10 measures. It is reasonable to assume that Liz Taylor is the proxy for Greg Whalley. Louise Kitchen is number three and is one of the most influential people at Enron where she pioneered on-line trading. Number four is Sally Beck, VP Energy Operations. Number five is Kenneth Lay, CEO of the Company. Overall, the list of the most influential employees underscores the importance of Trading, with eight representatives, Legal Department with four representatives, and Regulatory and Government Affairs with four representatives. There are five non-executive level employees on the list. One of them, Bill Williams, was implicated in energy price-fixing at Enron[13] and was managing the largest trading group shortly before the Enron's collapse[14].

Table 3.9 shows the top 10 employees in one of the centrality measures in the filtered attachments network. It is still dominated by high-level executives but there are also more regular employees. There are 10 employees from the communication network who held their position in one of the top 10 centrality measures but only three of them, Phillip Allen, Mike Grigsby, and John Lavorato, remained in the overall top 10 ranking. Communication network ranking is in parentheses next to the employee's name. Overall, traders dominate the ranks with 14 representatives out of which three traders are in the top five. This underscores the fact that Enron's main business is Energy trading. Next there are six representatives from the Legal Department, which shows that Enron had to address many legal issues as part of energy trading. This correlates with the fact that out of 15 new top employees in the attachments network, 11 are traders and 4 are from Legal Department.

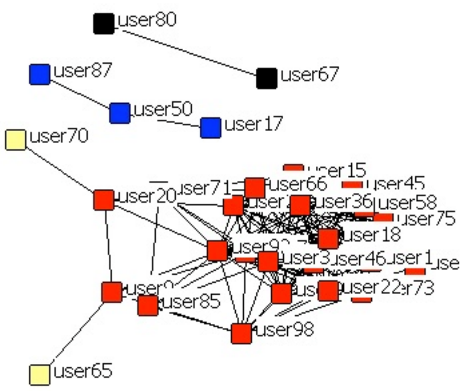
[13] `http://www.nytimes.com/2005/02/04/us/tapes-show-enron-arranged-plant-shutdown.html`
[14] `http://www.mresearch.com/pdfs/89.pdf`



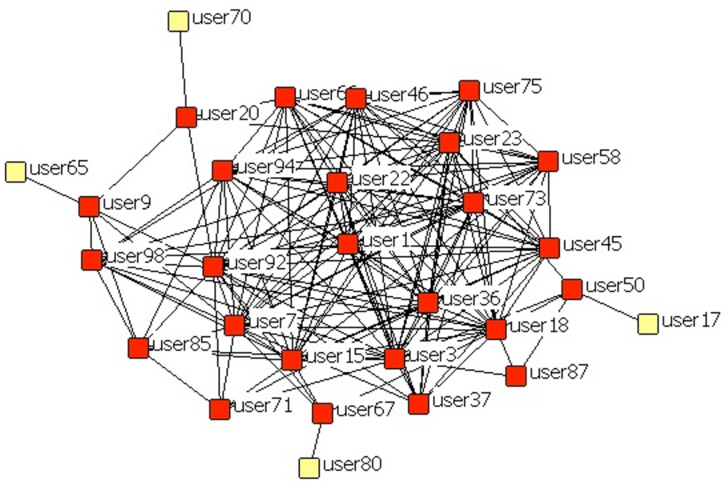
(a) Frequency >1.



(b) Frequency >2.



(c) Frequency >3.



(d) Frequency >10.

Figure 3.9: Attachments network with different filtered attachment's frequency in private archive.

This again stresses the influence of the Trading and Legal Departments within the Enron Organization. This also indicates Traders and Legal Department employees had to deal with substantial document handling as part of their responsibilities.

The rank in the curly brackets next to the employee name in Table 3.9 shows the overall ranking of the employee in the unfiltered attachments network. We see that employees move in both directions of overall ranking in filtered network as compared to unfiltered network and two employees Louise Kitchen {4} and Kevin Presto {5}, who are ranked high in the unfiltered network, have been removed from any top 10 ranking.

3.3.3 Gained and lost ties analysis

Besides affecting top 10 employees, the different definition of ties has, as expected, substantial impact on the overall number of ties in the network. There are 388 new ties and 704 lost ties in the attachments as compared to the communication network. I reviewed top 10 new ties in Table 3.10. The "Friend-of-a-friend" column shows a "friend" who contributed the most shared attachments that created a tie between two employees.

Table 3.10: Top 10 gained ties in attachments network.

Name	Name	Friend-of-a-friend	Ties
Stacy Dickson,SnrCnsl	Marie Heard, SnrLglSpcl	Tana Jones, SnrLglSpcl	128
Stacy Dickson,SnrCnsl	Elizabeth Sager, VP&AsstGenCnsl	Tana Jones, SnrLglSpcl	108
Keith Holst,Trader	Frank Ermis,Dir Trading	Mike Grigsby,VP Trd	20
Tori Kuykendall,Mgr Trading	Jason Wolfe,Trader	Mike Grigsby,VP Trd	18
Scott Hendrickson,Trader	Judy Townsend,Trader	Chris Germany,Mgr Trd	16
Randal Gay,Trader	Keith Holst,Trader	Mike Grigsby,VP Trd	16
Randal Gay,Trader	Jason Wolfe,Trader	Mike Grigsby,VP Trd	15
Matt Smith,Associate	Jason Wolfe,Trader	Mike Grigsby,VP Trd	15
Matt Smith,Associate	Randall Gay,Trader	Mike Grigsby,VP Trd	15
Jay Reitmeyer, Associate	Barry Tycholiz, VP Trading	Mike Grigsby, VP Trd	14

Employees have their ties due to a FOAF relationship where the friend has the same functional position as the employees. For instance, Stacy Dickson and Marie Heard are both from Legal Department, ENA. Their tie, or shared attachment, was created by Tana Jones, who sent a document to both of them; i.e., they both were on the *To, Cc*, or *Bcc* list.

Analysis of lost ties shows that in those cases employees have fewer shared attachments that are filtered out or no shared attachments at all. Also, out of 704 lost ties, 593 have communication frequency less or equal to five emails and 353 out 388 new ties have less or equal to five shared attachments. Consequently, if the lower bound threshold on the frequency of ties is set to five then there will be fewer lost and gained ties.

3.3.4 K-nearest neighbor analysis

The analysis above is a guesstimate in nature. Based on centrality measures of the extracted shared attachments network, I rate the top 10 most influential employees and attempt to corroborate my findings with Enron's organizational charts and on-line news stories. The extracted network is based on information sharing and reflects inter-group and intra-group interactions within functional teams and does not necessarily overlap with the organization charts. This type of analysis can be used by sociologists, anthropologists, or managers to improve communication efficiency within as well as outside of a company. Analysis of gained ties can be used to discover hidden relationships, which can not be discovered with the communication network. This is the use case I cover in section 3.2 when I discuss edge direction in the shared attachments network. Other type of analysis could be similar to Wang et al. [150, p2] where a Jaccard similarity matrix is generated and then clusters of user groups are derived from this matrix. According to the homophily principal in Mcpherson et al. [102], a contact between similar people occurs at a higher rate than amongst dissimilar people. Since the focus of the shared attachments network is on information sharing, the similarity and clustering approach may produce valuable insight into functional, organizational, and social group interactions. In this section I analyze the similarity of a subset of Enron employees based on the k-nearest neighbor algorithm. The idea is to use the employee's dictionary of an attachment's frequency as features for the k-nearest neighbor model, where an attachment is keyed by its SHA1. We can then query the model for the list of k nearest neighbors for an employee of interest.

Listing 3.1 shows the Python code example to generate the nearest neighbor list.

Listing 3.1: Nearest neighbor Python example.

```
import graphlab as gl
# shared.txt format: user name,
# attach1-sha1 attach2-sha1 ... attachN-sha1
data = gl.SFrame.read_csv('./shared.txt',
delimiter=',', header = True)
data['words'] = gl.text_analytics.count_words(
data['attachments'])
model = gl.nearest_neighbors.create(data,
features=['words'], label='name')
# query for 6 nearest neighbors of 'Kenneth Lay'
model.query(data[data['name'] == 'Kenneth Lay'],k=6)
```

I generated similarity lists for five Enron's employees: Kenneth Lay, Enron Chairman & CEO; Jeff Skilling, Enron President & COO; John Lavorato, Enron Americas COO, ranked number one in the communication network; Phillip Allen, Managing Director Trading, ranked number one in the attachments network; Stacy Dickson, ENA Attorney, has the most gained ties. The lists are shown in Table 3.11.

We see that within each similarity group, employees are either a direct report of the top employee in the group, the manager of the employee, the peer, the descendant in the same tree branch of the organizational chart, or appear to be part of a functional group. There are some interesting points about the lists. Rosalee Fleming is not listed in any of the organizational charts that I used for title reference. I found a reference to her title of CEO secretary in one of Enron's email. Her degree centrality measure ranks number 17 in Kenneth Lay's ego communication network. Clearly, as Kenneth Lay's secretary, she should have handled information exchanges between Kenneth Lay and other employees. The data fed from the attachments sharing network into the k-nearest neighbor model shows just that. Other points concern Matt Smith and Marie Heard. Neither of them has information available about their direct manager. The k-nearest neighbor model accurately predicts their similarity to ENA GW and Legal department, respectively, judging by other employees titles in the similarity groups. I analyzed their in/out degree for the person with whom they communicate the most and then by researching relevant emails and embedded organizational charts discovered their managers respectively as Keith Holst and Sara Shackleton. I find these results encouraging and they validate my approach to extracting social network via email's shared attachments.

It is a reasonable research question to ask if we can build a comparable k-nearest neighbors model for the communication network and evaluate attachments against this network. According to the homophily principal, contact between similar people occurs at a higher rate than amongst dissimilar people. Consequently, degree centrality is a good fit for generating a Jaccard similarity index. In this case the index is the fraction of emails exchanged between two users to all email communications of these two users. This could be done for overall degree, indegree, and outdegree. Empirically I find that indegree produces the best result. Therefore, the test point for each user is a vector of received emails from each user in the network. This is consistent with other research which demonstrates that the indegree is better at hierarchy detection in a social network Gupte et al.[72, p2], Michalski and Kazienko[108, p8]. Similarity lists for the same five

Table 3.11: Similarity lists based on the k-nearest neighbor model for attachments network. The last name in parenthesis is employee's manager.

Name	Title	Reporting to
Kenneth Lay	Enron Chairman&CEO	Board of Directors
Rosalee Fleming	Enron Chairman's secr	Kenneth Lay
Greg Whalley	EWS President&COO	Mike Frevert EWS Chrmn&CEO(Skilling)
James Derrick	Exec VP&GC	Jeff Skilling
Steven Kean	Exec VP&Chf of Staff	Jeff Skilling
Jeff Skilling	Enron President&COO	Kenneth Lay
Jeff Skilling	Enron President&COO	Kenneth Lay
Rick Buy	Exec VP&Chf Risk Offcr	Jeff Skilling
James Derick	Exec VP&GC	Jeff Skilling
Kenneth Lay	Enron Chairman&COO	Board of Directors
Greg Whalley	EWS President&COO	Mark Frevert, EWS Chrmn&CEO(Skilling)
David Delainey	EA President&COO	Greg Whalley
John Lavorato	EA COO	David Delainey
Louise Kitchen	EN President&CEO	Greg Whalley
David Delainey	EA President&COO	Greg Whalley
Greg Whalley	EWS President&COO	Mark Frevert, EWS Chrmn&CEO(Skilling)
Kevin Presto	ENA EP VP	John Lavorato
Jeffrey Shankman	EGM COO	Mike McConnell EGM Pres&CEO(Whalley)
Phillip Allen	ENA GW Mng Dir Trd	John Lavorato
Mike Grigsby	ENA GW VP Trdng	Phillip Allen
Keith Holst	ENA GW Dir Trdng	Vince Kaminski EWS Mng Dir(Lavorato)
Matt Smith	ENA GW Associate	Keith Holst
Matthew Lenhart	ENA GW Analyst	Mike Grigsby
Jane Tholt	ENA GW Dir Trd	Mike Grigsby
Stacy Dickson	ENA Sr Cnsl	Jeff Hodge ENA VP&AGC(Haedicke)
Tana Jones	NWFT Sr Lgl Spclst	Mark Taylor VP&GC(Haedicke)
Marie Heard	ENA Lgl Spclst	Sara Shackleton VP Networks&SC(Taylor)
Elizabeth Sager	ENA Power Trd AGC	Mark Haedicke EWS Mng Dir&GC
Mark Taylor	NWFT VP&GC	Mark Haedicke EWS Mng Dir&GC
Debra Perlingier	ENA Sr Lgl Spclst	Jeff Hodge ENA VP&AGC(Haedicke)

Table 3.12: Similarity lists based on the k-nearest neighbor model for communication network. The last name in parenthesis is employee's manager.

Name	Title	Reporting to
Kenneth Lay	Enron Chairman&CEO	Board of Directors
Jeff Skilling	Enron President&COO	Kenneth Lay
Greg Whalley	EWS President&COO	Mike Frevert EWS Chrmn&CEO(Skilling)
Rick Buy	Exec VP&Chf Risk Offcr	Jeff Skilling
James Derrick	Exec VP&GC	Jeff Skilling
Steven Kean	Exec VP&Chf of Staff	Jeff Skilling
Jeff Skilling	Enron President&COO	Kenneth Lay
Rick Buy	Exec VP&Chf Risk Offcr	Jeff Skilling
Kenneth Lay	Enron Chairman&CEO	Board of Directors
Steven Kean	Exec VP&Chf of Staff	Jeff Skilling
James Derick	Exec VP&GC	Jeff Skilling
Sally Beck	EA,VP Energy Operations	John Lavorato, EA COO
John Lavorato	EA COO	David Delainey
Louise Kitchen	EN President&CEO	Greg Whalley
Greg Whalley	EWS President&COO	Mark Frevert, EWS Chrmn&CEO(Skilling)
Jeffrey Shankman	EGM COO	Mike McConnell EGM Pres&CEO(Whalley)
David Delainey	EA President&COO	Greg Whalley
Andy Zipper	EWS, VP of Enron Online	Louise Kitchen
Phillip Allen	ENA GW Mng Dir Trd	John Lavorato, ENA COO
Mike Grigsby	ENA GW VP Trdng	Phillip Allen
John Lavorato	EA COO	David Dealiney
Keith Holst	ENA GW Dir Trdng	Vince Kaminski EWS Mng Dir(Lavorato)
Scott Neal	ENA GE VP Trdng	Hunter Shivelly, EA VP(Lavorato)
Fletcher Sturm	ENA EP, VP Trdng	Kevin Presto, ENA EP VP Trdng(Kitchen)
Stacy Dickson	ENA Sr Cnsl	Jeff Hodge ENA VP&AGC(Haedicke)
Susan Bailey	NWFT Sr Lgl Spclst	Mark Taylor VP&GC(Haedicke)
Stephanie Panus	NWFT Sr Lgl Spclst	Mark Taylor VP&GC(Haedicke)
Marie Heard	ENA Lgl Spclst	Sara Shackleton VP ENA&SC(Taylor)
Carol Clair	NWFT GCA	Mark Taylor VP&GC(Haedicke)
Debra Perlingier	ENA Sr Lgl Spclst	Jeff Hodge ENA VP&AGC(Haedicke)

employees as are generated in attachments network are shown in Table 3.12.

As we can see, Kenneth Lay and John Lavorato neighbors have only one neighbor who is different in communication and attachments networks (neighbors who differ are shown in bold font). But the neighbors are still members of the executive team. Jeff Skilling has two different neighbors between the networks, but again they are members of the executive team. Phillip Allen has three different neighbors between the networks but in the attachments network all neighbors are from the same department while in the communication the neighbors are represented by four departments. Stacy Dickson has three different neighbors between the networks but in both cases they all are members of the Legal department. A general conclusion is that members of the Executive and Legal teams communicated and distributed information within their ranks. This is consistent with Diesner et al. [51, p21].

It is interesting to ask to what extent filtering of the low value attachments affects the k-nearest neighbor model. The motivation for this question is an assumption that for people who have strong ties, the *noise* attachments should have low impact on the strength of the tie. To test this idea I compare the k-nearest neighbor model of the first neighbor on all employees in both unfiltered and filtered networks. In 68% of the cases the neighbor does not change. If I filter out attachments in bulk-email with list size higher than 35, then the number of cases with the same first neighbor in the unfiltered and bulk-email filtered networks goes up to 74%. In the private archive network these numbers are respectively 75% and 93% (bulk-email with list size higher than 12 is filtered). The numbers appear to be high and, if my assumption is correct, then the result is significant in that it might be possible to predict the closest first neighbor of a group of people in the email archive network. A naive approach would be to find the first closest neighbor of a group in the unfiltered network and filtered network with the low boundary filters applied and then find a subset of the group which has the same neighbor in both networks.

3.3.5 K-means clustering analysis

As another way of evaluating the attachments network, I classify employees with the k-means clustering algorithm. K-means is one of the simplest and wildly-used unsupervised learning algorithms that solves a clustering problem. I run the algorithm on weighted Jaccard distance $N \times N$ matrix where N is the number of core employees in the Enron corpus. The index for each pair of employees is calculated as one minus the number of intersection of attachments divided by the number of attachments in the union between two employees. K-means takes as an input parameter the number of clusters. I assume that information exchange is higher within an organizational unit whether it is a functional team or a department. Since the functional team information is not available, I guesstimate the number of clusters based on the average number of employees in a department. Based on Enron organizational charts, the average department size in Enron is 10 employees. Since there are 150 core employees in the Enron dataset, I set the number of clusters to 15. Table 3.13 shows the result of the clustering. Within each cluster I group employees by their respective department.

We see that employees in clusters 1, 6, 11, 12, and 14 are entirely within departmental boundaries. Cluster 9 consists of top level corporate executives, including Kenneth Lay (CEO) and Jeffrey Skilling (President). Clusters 8, 10, and 13 have only one employee who is not in the same department with the rest of the employees. Cluster 15 also has

Table 3.13: K-means clustering results for shared attachments network. Employees are grouped by the department within the cluster.

Cl. #	Cl. size	Department	# employees
1	9	ENA West Power Real Time	9
2	8	ENA Gas Central	2
		ENA Gas East	4
		ENA Gas Texas	1
		Energy Operations	1
3	40	EES	1
		ENA East Power	4
		ENA Gas Central	7
		ENA Gas East	4
		ENA Gas Financial	2
		ENA Gas Texas	1
		ENA Gas West	2
		ENA Legal	5
		ENA West Power	5
		ETS	1
		EWS	3
		Energy Operations	4
		Regulatory and Government Affairs	1
4	11	ENA East Power	1
		ENA Gas Central	1
		ENA Gas East	4
		ENA Gas Financial	3
		ENA Gas Texas	2
5	6	ENA Gas West	4
		ENA Legal	2
6	5	ETS	5
7	6	ENA East Power	4
		ENA Legal	1
		Energy Operations	1
8	11	ENA Gas West	1
		ETS	10
9	13	ENA Legal	1
		ETS	1
		EWS	6
		Enron	5
10	6	ENA West Power	5
		ENA West Power Real Time	1
11	5	ENA East Power	5
12	6	ENA Legal	6
13	13	ENA Gas Texas	1
		ENA Gas West	12
14	4	ENA East Power	4
15	6	ENA Legal	1
		Enron	1
		Regulatory and Government Affairs	4

only one employee who is not in the same department as other employees in the cluster. The employee from Enron department is Steven Kean, Exec VP&Chief of Staff. His responsibilities included Regulatory&Government Affairs. Clusters 2 and 4 have only one employee who is not in one of the ENA Gas departments. Clusters 5 and 7 have two employees who are not in the same department as the rest of the employees. Cluster 3 has the highest number of employees from 13 departments. The majority of employees in this cluster are from ENA Gas departments. Four out of five Legal department representative are from Financial and Gas trading groups from within the Legal organization. While we see that clustering has grouped many employees by their respective departments, it would not be correct to generalize this result as the ability of this approach to predict organizational units. The shared attachments network reflects information interactions between employees and clustering shows information exchange between functional units that happened to overlap in some cases with organizational units or departments. We can conclude that many departments have the majority of their communication either within the department or a subset of the department boundary. Top-level corporate management has its own clique, perhaps highlighting lack of top-down information interaction within the organization. The largest cluster (number 3) is the most diverse in terms of inter-department information exchange, and judging by the number of representatives from ENA Gas, is mostly involved in Gas trading with other departments providing necessary support.

Is there a k-means clustering for the communication network comparable to that for the attachments network? To find clusters, I calculate Jaccard index in the same way as in the k-nearest neighbor model for the communication network. The number of clusters is set to 15 as in the attachments network. Table 3.14 shows results of the clustering. Clustering in the communication network appears to perform better in that clusters are more centered around departments with higher number of employees as can be seen in Table 3.15. Another observation is that clusters 4, 6, 12 in communication and 13, 8, 9 in attachments networks have a high number of overlapping employees in departments ENA Gas West, ETS, and Corporate with 11, 9, and 10 employees, respectively. This means employees in these departments have more communication and information interactions within their circle than outside. For Corporate employees these results correlate with the k-nearest neighbor model conclusion in section 3.3.4.

3.3.6 Department and title ranking analysis

Besides comparing the most influential employees in the communication and attachments networks, we can also look at department and title ranking by aggregating employees by department and by title. Tables 3.16 and 3.17 show the ranking for department and title, respectively.

We see that, at the department level, both networks have six departments ranked at the same level and both identify ENA Legal and Corporate as the two top-ranked departments. We see more consistency in terms of communication and information exchange at the department level. There is no overlap in top 10 ranking between communication and attachments network at the title level. But some titles like Director and Dir Trading differ only in one rank and Mgr Trading, President & CEO, VP, VP Trading, VP&Asst Gen Cnsl differ in two ranks.

Another comparison point between the communication and attachments networks are

Table 3.14: K-means clustering results for communication network. Employees are grouped by the department within the cluster.

Cl. #	Cl. size	Department	# employees
1	4	ENA West Power Real Time	4
2	18	ENA Gas Central	7
		ENA Gas East	11
3	9	ENA Gas Financial	5
		ENA Gas Texas	3
		EWS	1
4	17	ENA East Power	1
		ENA Gas Texas	1
		ENA Gas West	15
5	18	ENA East Power	16
		EWS	1
		Energy Operations	1
6	15	ETS	15
7	3	ENA West Power	2
		ENA West Power Real Time	1
8	15	ENA Gas Central	1
		ENA Gas East	1
		ENA Gas West	1
		ENA Legal	12
9	6	ENA East Power	1
		ENA West Power	4
		ENA West Power Real Time	1
10	11	EES	1
		ENA Legal	2
		ENA West Power	1
		Enron	2
		Regulatory and Government Affairs	5
11	3	ENA Gas Central	1
		ENA Gas Texas	1
		ENA Legal	1
12	15	ENA Legal	1
		ETS	2
		EWS	7
		Energy Operations	1
		Enron	4
13	8	ENA Gas Central	1
		ENA Gas West	3
		Energy Operations	4
14	4	ENA West Power	3
		ENA West Power Real Time	1
15	4	ENA West Power Real Time	4

Table 3.15: Clustering around departments in communication vs. attachments networks.

Department	Communication		Attachments	
	Cl. #	Cl. Size	Cl. #	Cl. Size
ENA West Power Real Time	1	4	1	9
ENA Gas East	2	11	2	4
ENA Gas West	4	15	13	12
ENA East Power	5	16	3	4
ETS	6	15	8	10
ENA Legal	8	12	8	10

Table 3.16: Department ranking in communication and attachments network.

Rank	Communication network	Attachments network
1	ENA Legal	ENA Legal
2	Corporate	Corporate
3	Regulatory & Government Affairs	ENA Gas West
4	ENA Gas East	ENA Gas East
5	ENA East Power	ENA East Power
6	ENA Gas West	ENA Gas Central
7	Energy Operations	Regulatory & Government Affairs
8	ENA Gas Central	Enron Wholesale Services
9	Enron Wholesale Services	ENA Gas Texas
10	ENA Gas Texas	ENA Gas Financial
11	Enron Transportation Services	Energy Operations
12	ENA West Power	Enron Transportation Services
13	ENA Gas Financial	ENA West Power
14	ENA West Power Real Time	ENA West Power Real Time
15	Enron Energy Services	Enron Energy Services

Table 3.17: Title ranking in communication and attachments networks.

Rank	Communication network	Attachments network
1	VP & Gen Cnsl	Director
2	Director	Mgr Trading
3	VP Trading	VP & Gen Cnsl
4	Mgr Trading	Dir Trading
5	Dir Trading	VP Trading
6	COO	President & CEO
7	Specialist Legal	VP
8	President & CEO	VP & Asst Gen Cnsl
9	VP	VP of Gov Affairs
10	VP & Asst Gen Cnsl	Analyst

the pairs of employees interacting the most within their respective ego networks. Perhaps employees who communicate the most, also exchange the most information. This does not appear to be the case within the Enron organization. Only 42.95% of pairs overlap between the communication and attachments networks. This type of interaction is organization-specific and does not necessarily point towards some dysfunction and can be explained by varying job responsibilities within each department. Indeed, as we see from the centrality measures ranking in Table 3.8 and Table 3.9, the most central employees in the communication network are top level executives, while in the attachments network Traders are the most central.

3.4 Private email archive network extraction

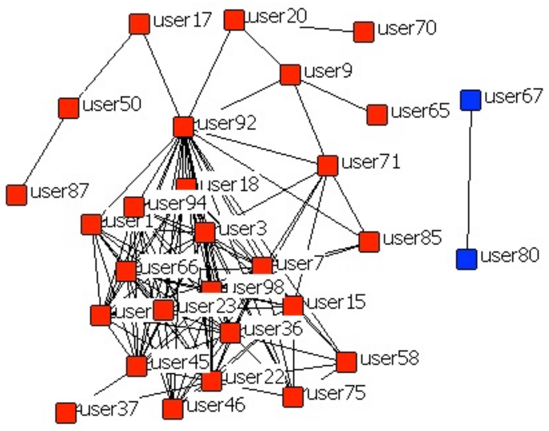
Just as in the Enron case, I extract two networks from the private archives. First is via communication between users, where nodes represent the email address or the user in *From, To, Cc*, and *Bcc* email header fields. Edges represent the relationship between the sender (*From*) and recipients (*To, Cc, Bcc*). Edges are undirected and weighted by the frequency of communication. Second is via shared attachments by constructing one-mode projection graph on users, where nodes represent users and edges represent attachments shared between the users. Tie strength is determined the same way as in the Enron's attachments network.

The low boundaries for filtering out attachments are set as outlined in section 3.2 to 10 KB for attachment size, 2 for the frequency of attachments in unique messages, 2 for the frequency of unique senders, and 12 for bulk email.

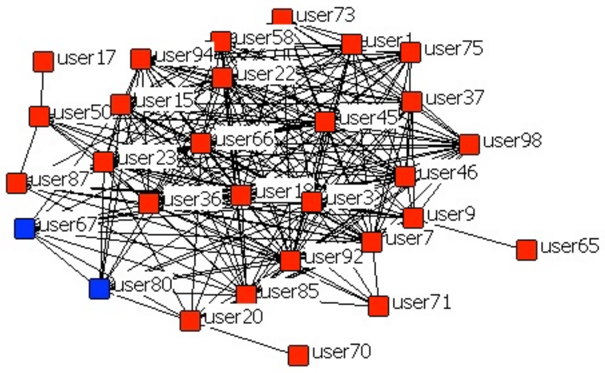
3.5 Private email archive network analysis

3.5.1 General statistics

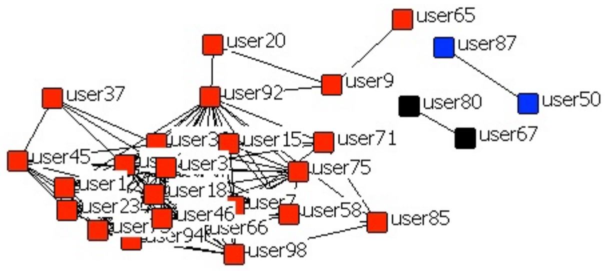
Table 3.18 shows general statistics for the communication and attachments networks of the private archives. We see some similarity between the private and Enron networks. The communication network in the private archive has a clustering coefficient 9%, network centralization 3.5% and network heterogeneity 18% higher than Enron's one; but, the density is almost twice as high in the private archive communication network. Findings point to a more densely connected network. This is not surprising considering that the private network is much smaller as compared to Enron, essentially a clique. The higher diameter, radius, and number of components in the private network is explained by emails from my work-related friends and members of their family who are not connected to my family and non work-related friends. Filtered attachments networks are less similar with clustering coefficient 12%, network centralization 44%, density 157%, and heterogeneity 14% higher than the corresponding Enron network. Again, it is a more densely connected and more centralized network than Enron's one. Finally, the unfiltered attachments network has clustering coefficient 7% and density 44% higher and network centralization 20% and heterogeneity 10% smaller than the respective Enron network. The reduction in two statistics can be explained by all components joining into one component and creating more leaf nodes in unfiltered attachments network as shown in Figure 3.10b.



(a) Communication network.



(b) Unfiltered attachments network.



(c) Filtered attachments network.

Figure 3.10: Communication and unfiltered/filtered attachments networks.

Within the private archive we see that the unfiltered attachments network has higher clustering coefficient and network density, smaller network diameter, and one component as opposed to two in communication network. This is explained by more connections being created due to attachment sharing. Network centralization and heterogeneity, however, are smaller in attachments network than in communication network. This is explained by FOAF relationships in the communication network as demonstrated in Figure 3.10a, where *user92* is a hub for seven other users connected through it to the rest of the network. However, in the unfiltered attachments network this is no longer the case as shown in Figure 3.10b. What is quite remarkable is that *user67* and *user80* who are isolated from the rest of the network in the communication network, join into one component in

Table 3.18: General statistics for private communication and attachments networks.

Statistics	Communication network	Attachments network, filtered	Attachments network, unfiltered
Clustering coefficient	0.596	0.634	0.767
Connected components	2	3	1
Network diameter	5	4	4
Network radius	3	2	2
Network centralization	0.464	0.414	0.324
Characteristic path length	1.986	1.632	1.525
Avg. number of neighbors	8.897	9.037	16.552
Number of nodes	29	27	29
Number of edges	54 731	5 262	11 324
Network density	0.318	0.348	0.591
Network heterogeneity	0.696	0.646	0.435

the attachments network and have degree greater than 10. This is due to the fact that they have attachments shared with other users in the network. My email archive has four attachments shared with these users therefore I can visually review those attachments. All of the attachments are emoji images of sizes roughly from 0.5 KB to 4 KB. Consequently, those attachments are filtered out by the 10 KB attachment size threshold. This case underscores the need for a comprehensive attachment filtering approach. The filtered attachments network statistics is bounded by communication and unfiltered attachments network statistics. The clustering coefficient is 6%, density 9% higher and network centralization 11% and heterogeneity 7% lower than in communication network. Filtering removed almost half the edges in the filtered attachments network as compared to the unfiltered one and also splits *user87* and *user50* into a separate component. But there are seven nodes in communication network as opposed to five nodes in attachments network with clustering coefficient 0. This explains the higher clustering coefficient and density in the filtered attachments network than in the communication network. The second two characteristics are explained by the reduced hub node *user92* neighborhood. The average number of neighbors in filtered attachments network is only 1.5% higher than in communication network. Two nodes *user17* and *user70* are removed from filtered attachments network. *user17* is connected to the rest of the network through the hub node *user92* (Figure 3.10a) and only has shared attachments with *user50* (Figure 3.10b). After the filtering, *user17* does not have any shared attachments with *user50* and is removed from the network (Figure 3.10c). *user70* is connected to the rest of the network via hub *user20* in both communication and unfiltered attachments networks. After the filtering, *user70* loses her shared attachments to *user20* and is removed form the network. Overall, we see that filtered attachments network is similar to the communication network in their characteristics.

3.5.2 Centrality measures analysis

I calculate degree, eigenvector, betweenness, and closeness centrality measures for communication and filtered attachments network. Results are shown in Tables 3.19 and 3.20.

Table 3.19: Top 10 users in centrality measures in private communication network.

Overall Rank	Name	Centrality Measures
1	user92	D(3),EV(1),B(1),C(1)
2	user7	D(1),EV(4),B(7),C(4)
3	user36	D(5),EV(2),B(9),C(3)
4	user22	D(10),EV(3),B(6),C(2)
5	user3	D(4),EV(5),B(12),C(5)
6	user18	D(6),EV(9),B(14),C(7)
7	user23	D(9),EV(8),B(15),C(8)
8	user15	D(2),EV(13),B(13),C(10)
9	user98	D(13),EV(6),B(11),C(6)
10	user66	D(11),EV(7),B(16),C(9)
11	user94	D(8),EV(10),B(17),C(12)
12	user17	D(28),EV(23),B(2),C(20)
13	user9	D(22),EV(21),B(3),C(19)
14	user20	D(21),EV(22),B(4),C(21)
15	user45	D(14),EV(11),B(8),C(11)
16	user46	D(7),EV(15),B(20),C(15)
17	user50	D(23),EV(26),B(5),C(24)
18	user71	D(20),EV(18),B(10),C(17)

While the ranking shows the most central users in the private communication and attachments networks, it is wrong to make an inference about a user's influence. First, this is a small network of only 29 users. Second, this is essentially my ego network. And, last but not least, my opinion is biased and subjective. Consequently, it suffices to say that exactly for the second reason (the ego network), I am ranked number one in the list followed by people who are closest to me in my family. In terms of the evaluation of the attachments network, the list of gained and lost ties, k-nearest neighbor model, and clustering provide more insight and opportunity for generalization.

3.5.3 Gained and lost ties analysis

To describe the relationship between users in the private network, I use the categories shown in Table 3.21. These groups are based on Hill et al. [77, p56].

Table 3.22 shows ties gained in the attachments network. The "Friend-of-a-friend" column shows a "friend" who contributes the most shared attachments that create a tie between two users. I show in parenthesis the relationship between two users. For instance, the tie between *user37* and *user36* is due to an attachment sent from *user45* to users *user37* and *user36*. *user45* and *user37* are husband and wife and *user36* is a friend of *user45*. This is a typical case of FOFA relationship. *user58* and *user3* have a "relative affinal distant relatedness" type of relationship. The tie between them is due to different attachments sent by *user7, user15, user36,* and *user92*. There is a "relative genetic relatedness" or "affinity relatedness" between these users and *user58* and *user3*. In all described cases tie establishment, with the exception of one, can be characterized as an FOFA relationship. The exception is *user75* and *user66* who have "relative affinal distant relatedness". The distribution pattern of this attachment is shown in Figure 3.11. *user22*

Table 3.20: Top 10 users in centrality measures in private attachments network.

Overall Rank	Name	Centrality Measures
1	user92(1){3}	D(4),EV(3),B(1),C(1)
2	user3(5){4}	D(3),EV(1),B(3),C(2)
3	user7(2){2}	D(1),EV(5),B(5),C(5)
4	user22(4){8}	D(7),EV(2),B(6),C(3)
5	user36(3){7}	D(6),EV(4),B(4),C(4)
6	user15(8){12}	D(2),EV(9),B(10),C(8)
7	user18(6){4}	D(5),EV(6),B(7),C(6)
8	user94(11){16}	D(12),EV(7),B(8),C(7)
9	user66(10){6}	D(10),EV(8),B(13),C(9)
10	user9(13){10}	D(21),EV(21),B(2),C(20)
11	user73	D(16),EV(10),B(11),C(11)
12	user46(16){14}	D(13),EV(11),B(15),C(10)
13	user58	D(8),EV(12),B(19),C(14)
14	user75	D(9),EV(13),B(20),C(15)
15	user98(9)	D(15),EV(16),B(9),C(16)

Table 3.21: User relationship categories in private network.

Relation type	Abbreviation	Example of the relationship
Relative genetic relatedness	Rg	Mother and daughter
Relative affinal relatedness	Ra	Brother in-law
Relative affinal distant relatedness	Rd	Distant relatives, meet at parties
Friend	F	Friends, keep in touch once a month
Acquaintance	A	Occasionally meet at parties
Work colleague	W	Office co-workers but not friends
None	N	There is no relationship

Table 3.22: Gained ties in private attachments network.

Name	Name	Relation	Friend-of-a-friend	Ties
user58	user3	Rd	user7(Ra,Rg),user15(Rg,Ra),user36(Ra,Rg),user92(Ra,Ra)	28
user37	user36	A	user45(Ra,F)	5
user58	user18	Rd	user15(Rg,Ra),user36(Ra,Rg)	2
user75	user66	Rd	user15(Rg,Ra):user22(Ra,Ra)	1
user58	user46	Rd	user7(Ra,Rg)	1
user94	user15	Rd	user7(Ra,Ra)	1
user7	user1	A	user92(Rg,F)	1
user46	user3	Ra	user7(Rg,Rg)	1
user37	user3	A	user22(A,Ra)*	1
user66	user58	Rd	user22(Ra,Ra)*	1
user75	user46	Rd	user7(Ra,Rg)	1

sent an attachment to *user3, user7, user15, user66* and *user92* and *user15* forwarded this attachment to *user58* and *user75.* This is a somewhat unusual case in that the identical email message with the shared attachment exists in *user3, user7, user15, user22, user66* and *user92* email archives. The message does not have *To, Cc, Bcc* headers set and the value in the *From* header is *user22. user58* has the email forwarded to her by *user15. user75* does not have an email with this attachment in her archive. This underscores the importance of the tie inference when extracting the network not just explicitly from email headers but also implicitly by associating the attachment with the email archive's owner. Indeed, if we use just the email headers to infer the attachment sharing then, due to the lack of the *To, Cc, Bcc* headers, we can only say that *user22, user15, user58* and *user75* share the same attachment. If we also use implicit inference, then we can assert the relationship between all users in this case. It is worth noting that the cases covered in Table 3.22, have a weak tie, namely with the exception of {*user46, user3*}, the rest of the ties are either "acquaintance" or "relative affinal distant". This makes sense as we expect people with strong ties to have consistent communication and information exchange.

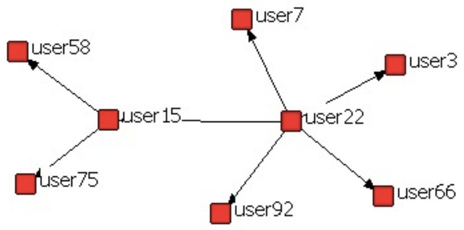


Figure 3.11: Gained tie trace for user75 and user66.

Table 3.23 shows lost ties in attachments network. *user98* has the majority of lost ties. *user98* is what I call a "membership account" for *user36.* "membership account" is probably a common trend where a user opens an email account for various e-site membership so that machine generated messages are sent to this account rather than to her main email account that is used for communication with family, friends, or business. We see more diversity in the type of tie relation in the case of lost ties. It would be a fair assessment of lost tie cases to say that they are lost because they are weak ties. Indeed, most of these ties have low communication exchange - less or equal to seven. It is reasonable

Table 3.23: Lost ties in private attachments network. Last column is for communication network.

Name	Name	Relation	Number of ties
user98	user23	Ra	37
user92	user45	A	30
user98	user66	Ra	7
user73	user23	A	5
user98	user15	Ra	5
user98	user36	Same user	5
user94	user66	Ra	5
user66	user45	A	4
user70	user20	Ra	4
user9	user71	F	3
user98	user1	F	3
user7	user45	A	2
user23	user1	A	2
user50	user17	Rg	2
user23	user15	Ra	2
user92	user17	W	2
user98	user45	F	1
user66	user46	Ra	1

to expect that a low communication exchange has a low chance of having any email with attachments let alone shared attachments. The "membership account" of *user98* makes {*user98, user23*} tie a weak tie. And {*user92, user45*} has an "acquaintance" type of relationship, which is a weak tie too. We can conclude that both gained and lost ties in the attachments network as compared to the communication network, are weak ties.

3.5.4 K-nearest neighbor analysis

Table 3.24 shows results of the k-nearest neighbor model for each user. Results are grouped by the user's relationship and only unique pairs are selected; i.e. a pair *userA, userB* and pair *userB, userA* are counted once. Out of 27 pairs, 20 pairs have symmetrical neighbor prediction, where *userB* is the nearest neighbor of *userA* and *userA* is the nearest neighbor of *userB*. Seven pairs with asymmetrical neighbor prediction have the nearest neighbor within a close social circle of friends or family. For instance, *userB* is the nearest neighbor of *userA* and their relation is friends, where *userC* is the nearest neighbor of *userB* and their relation is a child and a mother.

I pose the same question here of comparable k-nearest neighbor model for communication network as I do for the Enron data corpus in Section 3.3.4. Unexpectedly, the model performs poorly. For example, the ommunication network has 6 versus 20 symmetrical pairs in the attachments network. Three predictions out of six are not the closest possible relation. For example one pair has "Acquaintance" relation when the best relation is "Husband & Wife" and "Parent & Child". It is not likely that the reason for the poor performance is relatively the low number of users in the private as compared to Enron networks because the model performs well in the private attachments network. It is most

Table 3.24: Similarity lists based on the k-nearest neighbor model in private attachments network. Pairs are grouped by relation.

Relation	Count
Husband & Wife	6
Parent & Child	4
Siblings	4
Same user	2
Friends	1
Co-workers	1

Table 3.25: K-means clustering results. Clusters are grouped by relation.

Relation	Number of clusters
Husband & Wife	5
Siblings	3
Parent & Child	1
Same user, two accounts	1
Single	2
Friends	1
Husband & Wife + Parent	1

likely that the Jaccard index based on indegree is not a good fit in this case.

3.5.5 K-means clustering analysis

Table 3.25 shows k-means clustering results. Clusters are grouped by a user's relation. I choose 14 clusters, which roughly correspond to each cluster having two users. Eight clusters (Husband & Wife and Siblings), correspond to symmetrical pairs and one cluster (Husband & Wife) corresponds to asymmetrical pair in Table 3.24. All clusters, with the exception of two clusters having one user each, are within a close social circle of friends or family, just as in the case of the k-nearest neighbor.

Just as in the k-nearest neighbor model in section 3.5.4, k-means clustering for the communication network performs poorly with 10 clusters having only one user, 4 clusters having three, three, four, and eight users, respectively, and only 1 cluster having two users each grouped correctly according to their social relation.

3.6 Flickr OSN network analysis

I analyze the Flickr dataset to demonstrate that the SNA of the network extracted from the documents shared between users is applicable not only to email archives but also to OSN.

Flickr allows users to share photos privately or publicly, create and join groups of interest, start a topic discussion thread within the group, create a list of friends as contacts, annotate photos with comments and tags, and mark favorite photos. These are just a few common functionalities of Flickr OSN. Flickr provides REST API and third party libraries for popular programming languages to access its server[15]. Flickr datasets have been extensively used in SNA research. Rae et al. [122] combines information from all photos in the system, user's personal photos, user's social contact photos, and photos posted in the user's group for tag recommendation. Valafar et al. [146] analyze interractions between the photo's owner and fans. The authors show that only a very small fraction of users are active as owners and fans - 30% of the top 1 000 fans are among the top 1 000 owners and that the highest level of reciprocation of 28% occurs among the top 1 000 fans and owners. Alves et al. [18] analyze how users are affected by their social network and how they affect the network in which they are included. The authors look at proportions of photos marked as favorite by a user in her network and the proportion of this user's photos marked as favorite by other users in the network. The results show that 70% of favorite photos are marked by user's contacts and 20% by users at the distance 1 and 2. This demonstrates that users prefer content posted by their contacts and contacts of other users with whom they are connected. Cha et al. [38] use favorite photos with time stamps and contact lists to analyze social cascades. They conclude that the social network plays a significant role in information dissemination.

I extract a network from the Flickr dataset by connecting all users sharing the same image via favorite images. For instance, if a user *U1* is the owner of the image *G* and users *U2* and *U3* mark *G* as their favorite photo, then there are the following edges {*U1, U2*}, {*U1, U3*}, {*U2, U3*} connecting these users. The resulting network has 2 000 177 nodes with the power-law node degree distribution characteristic of OSN as can be seen from Figure 3.12. In order to reduce the computational load required for structural analysis, I extract a subgraph with nodes of degree greater than 30 000 from this network. The extracted subgraph has 1 231 nodes and 631 115 edges. Figure 3.13 shows the subgraph degree distribution. It no longer follows a power-law distribution because the nodes with degree less than 30 000 are removed, which reduces the degree of other nodes. Lescovec et al. [95] analyze over 100 large sparse real-world social and information networks to evaluate community detection, evaluation and typical sizes. They conclude that the best communities include up to 100 nodes, that the best communities gradually blend in with the rest of the network and become less community-like, and that large communities can be broken into smaller ones, each of which is more community-like than the original supposed community. Dunbur et al. [57] analyze OSN ego networks and conclude that they follow the same layered structure of 5, 15, 50, and 150 sized groups as offline networks. Kumar et al. [90] demonstrate segmentation of OSN into three regions of singletons not participating in the network, isolated star structure communities, and a giant component anchored by the well-connected core region. Consequently, I assume that the extracted subgraph with 1 231 nodes should have at least 12 communities with about 100 number of nodes each.

3.6.1 General statistics

Table 3.26 shows the general statistics of the core subgraph with nodes of degree greater than 30 000. As expected, the statistics shows a densely connected core of nodes. The interconnecting of fans of the favorite picture result in a high clustering coefficient, density, and an average number of neighbors and in low diameter, centralization, and heterogeneity.

[15] `https://www.flickr.com/services/api/`

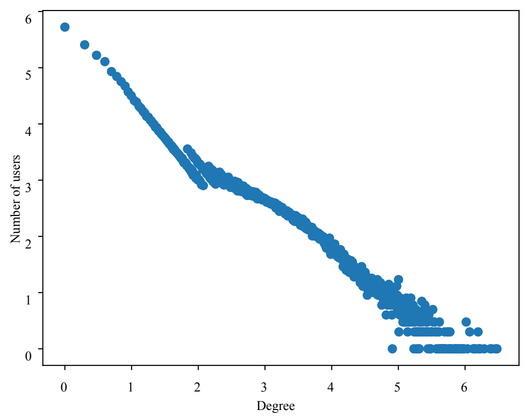


Figure 3.12: Degree distribution for the Flickr crawled dataset (log-log scale).

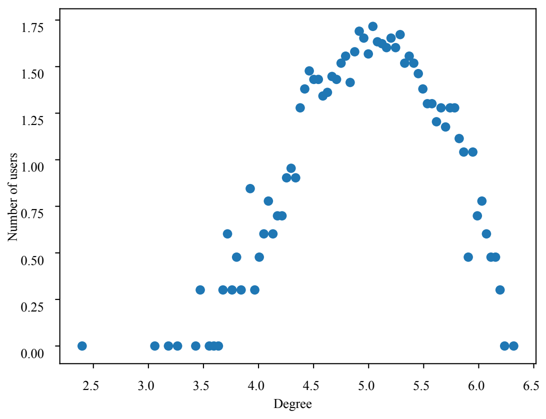


Figure 3.13: Degree distribution for the Flickr core subgraph with the nodes' of degree greater than 30 000 (log-log scale).

3.6.2 K-means clustering analysis

I cluster the users in the core subgraph into 12, 24, 82, and 246 clusters, respectively. Since there are 1 231 users in the subgraph, the number of users in each cluster is roughly 100, 50, 15, and 5. This corresponds to layered structures of 50, 15, 5 as noted in Dunbar et al. [57] and the best community size of 100 in Lescovec et al. [95]. I evaluate each clustering result by analyzing the average pairwise Jaccard similarity between users in each cluster. Similarity is based on a user's groups, contacts, and images. The rationale behind this is that users with common interests interact more. Consequently, the expectation is that users within a cluster have higher membership in the same groups, contacts, or are fans of the same photos at a higher rate. Since I cluster users from a densely connected subgraph, as can be seen from the clustering coefficient and the network density in Table 3.26, it is possible that if we cluster users randomly it may have similar results in terms of the high rate of user's membership in groups, contacts, or sharing the same images. To test this hypothesis I compare user's similarity in k-means clusters with the average user's similarity in five randomly selected clusters. Table 3.27 shows results of the clustering. We see that in both k-means and random clusters the Jaccard similarity is small. This is

Table 3.26: General statistics for the Flickr photo-sharing network sample - subgraph of users with the number of nodes greater than 30 000.

Clustering coefficient	0.906
Connected components	1
Approx. full diameter	2
Network centralization	0.166
90% shortest path length distribution	1.53
Avg. number of neighbors	1 025
Number of nodes	1 231
Number of edges	631 115
Network density	0.833
Network heterogeneity	0.197

explained by high number of groups, contacts, or images that the user is the member or a fan of. Indeed, in the core subgraph a user in average is the member of 444.52 groups (*CV* 1.47), has 2 875.70 contacts (*CV* 3.0), or is a fan of 12 512.44 images (*CV* 1.29). We can make the following observations from the Table 3.27:

- Similarity for images is higher than for groups or contacts within the cluster. This is expected since the clustering is done on the image's similarity matrix.

- Image-based similarity has an upward trend as the number of clusters increases and consequently the size of the cluster decreases. This is expected behavior as well. Dunbar et al. [57] note that individuals do not distribute their social effort evenly among the alters in their network and that alters can be ranked in order of declining investment by ego with the ranking falling into natural series of layers of around 5, 15, 50 and 150 alters. We see that the smaller the group size the more interactions we can expect within this group.

- Contacts-based similarity has a slight upward trend as the number of clusters increases. This reinforces the point above.

- Groups-based similarity does not follow the upward trend of images and contacts similarity. This can be explained by the exclusivity of the contacts list, which represents the list of friends, and the shared images representing favorite photos. While on average the contacts list has 2 875.70 friends and a user on average is the fan of 12 512.44 photos, the average number of users in a group is 1 143.05 (*CV* 4.0), and with the average number of groups per user equal to 444.52 this creates a network of $1\ 143.05 * 444.52 = 508\ 108.58$ users.

- Similarity of random clusters within the same similarity type is constant; i.e., it does not depend on the cluster size. Moreover, it is inversely proportional to the perceived similarity's exclusivity; i.e., the more exclusive image's similarity has the lowest random cluster similarity.

Overall the k-means clustering based on image sharing between users has similarity 1.8-12.75 times higher than random clusters. Each random cluster is generated five times. We can then state the null hypothesis as the similarity based on image sharing, groups or

Table 3.27: User's similarity comparison in k-means clusters and random clusters.

Number of clusters	Clustering type	Similarity type	Similarity
12	K-means	Groups	0.026
12	Random	Groups	0.014
12	K-means	Contacts	0.022
12	Random	Contacts	0.007
12	K-means	Images	0.032
12	Random	Images	0.004
24	K-means	Groups	0.033
24	Random	Groups	0.014
24	K-means	Contacts	0.023
24	Random	Contacts	0.007
24	K-means	Images	0.039
24	Random	Images	0.004
82	K-means	Groups	0.029
82	Random	Groups	0.014
82	K-means	Contacts	0.023
82	Random	Contacts	0.007
82	K-means	Images	0.04
82	Random	Images	0.004
246	K-means	Groups	0.031
246	Random	Groups	0.014
246	K-means	Contacts	0.025
246	Random	Contacts	0.007
246	K-means	Images	0.051
246	Random	Images	0.004

contacts in k-means derived clusters is equal to the corresponding similarity in randomly derived clusters. The highest $p-value$ in each T-test is 10^{-6}. We can then reject the null hypothesis with confidence level 0.01 and conclude that the k-means clustering based on the user's favorites photo sharing finds distinctive clusters of users in Flickr.

3.7 Application of attachments SNA to energy savings analysis in the decentralized email architecture

3.7.1 Methodology

This Section demonstrates how discovering social relationships between users allows us to architect a more efficient system in terms of energy usage. The key to energy cost optimization is the backup strategy or the redundancy plan that takes advantage of duplicate data-sharing within a group of users. The idea is to use the homophily principal inherent in a social network; i.e., contact between similar people occurs at a higher rate than among dissimilar people, which means similar people tend to share more information with each other. Consequently, if we group users devices based on users similarity, where similarity is defined as information sharing, we can architect a more efficient system minimizing the number of required backup replicas. This idea is demonstrated in Figure 3.14. Let us assume that the backup strategy involves two redundant devices for the user to maintain replicas of a file in a group consisting of three users. The group is selected in a way that maximizes file sharing between two users and each device backs up files from the other device. I term this group the backup group. The backup group consists of devices where each device contains one replica of the file. There is generally a one-to-one relation between the user and the backup device. Let us assume that an email message consists of the email body B and the email attachment A. Let us further assume that energy consumption has two components to it: Erd - an energy unit to receive from the network and write one byte to the disk, Eft - an energy unit to fetch and transmit one byte over the network. For demonstration purposes, all devices have the same Erd and Eft. In Figure 3.14 (a), *User* receives the message, which consequently is replicated to other user's devices. The energy cost in this case is $E = (B + A) * Erd * 3 + (B + A) * Eft * 2$. In Figure 3.14 (b), *User* receives the message, but instead of replicating the message to the user's devices, the message is replicated to one device of *User1* and one device of *User2*. *User1* has an email message with the body $B1$ and the same attachment A that needs to be replicated by *User*. Because A is already stored by *User1*, it does not have to be replicated. In this case, *User* energy cost by sharing the devices for replication with *User1* and *User2* is $Es = E - A * (Erd + Eft)$. The energy savings can be defined as $S = 1 - (E - Es)/E = Es/E$, or

$$S = \frac{A \times (Erd + Eft)}{(B + A) \times (Erd + Eft) \times 2 + (B + A) \times Erd} \tag{3.1}$$

Divide both parts of the fraction by $(Erd + Eft)$ and assume that $Erd/(Erd + Eft)$ is equal to coefficient c, where c signifies the efficiency of receiving one byte of data versus the total energy of receiving and transmitting one byte of data. When Erd is equal to Eft, c is equal to 0.5. The equation for the savings is

$$S = \frac{A}{(B + A) \times (2 + c)} \tag{3.2}$$

Dividing both parts of the fraction by A, the equation for the energy savings is

$$S = \frac{1}{(\frac{B}{A} + 1) \times (2 + c)} \tag{3.3}$$

Equation 3.3 can give us an estimate of the expected energy cost savings. We can see that the smaller the ratio of the email's body to the attachment size, the higher the energy saving. Let us assume that the ratio is 0; for example the message only contains an attachment. In that case, cost saving depends on the coefficient c. If we assume that c is equal to 0.5, then the expected energy saving is 40%. We can also see that the smaller coefficient c is, the higher the energy savings is. Higher values of c indicate the energy cost of fetching from the disk and transmitting one byte of data is higher than the energy cost of receiving and writing one byte of data to the disk. Let us assume that c is close to 0, then energy savings is 50%. Naturally, 50% is an unrealistic case but it does establish the high boundary for energy cost savings. Another important observation from equation

3.3 is that the highest energy cost comes from the required number of replications. In the equation 3.3, the value 2 is the number of required replicas minus 1. If the number of required replicas is 6 then the energy saving with c equal to 0.5 and with B/A ratio close to 0 is 18%.

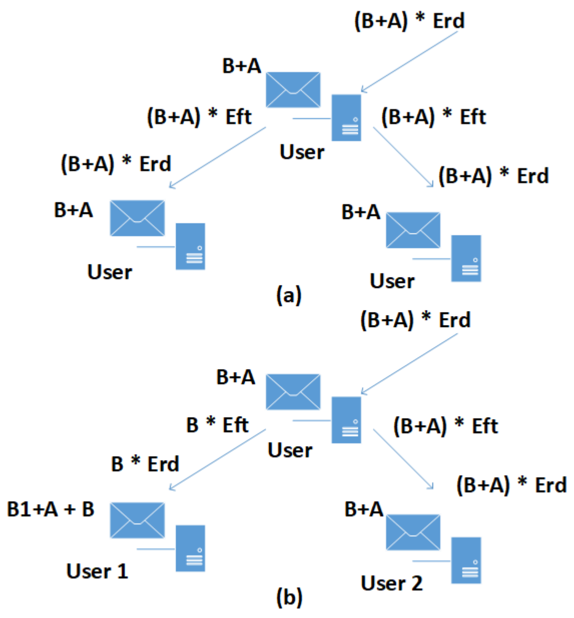


Figure 3.14: Energy saving demonstration. (a) User replicates the message to her devices. (b) User replicates the message to User1 and User2 devices.

Let us extend equation 3.3 to a general case. Let us assume that:

- An email message is received by a user's home IoT device with a globally accessible address and then is distributed to other devices. Let us term the device with the globally accessible address the hub. Other devices may include the user's own device or Cloud storage.
- R is the number of required replicas of the email message.
- Energy consumption when receiving from the network and writing to the disk one byte of data by the hub is Erd.
- Energy consumption when fetching from the disk and transmitting to the network one byte of data by the hub is Eft.
- Energy consumption when receiving from the network and writing to the disk one byte of data by another device is Erd_k, where k is in $\{1, 2, ..., (R-1)\}$.
- An email message is composed of the message body B_{ui} and attachment A_{uj}, where u is in $\{1, 2, ..., U\}$, U is the number of users, i is in $\{1, 2, ..., N\}$, N is the number of email messages in the user's archive. j is in $\{1, 2, ..., M\}$, M is the number of email attachments in the user's archive and A_{uj} could be equal to 0.
- G_g is the number of users in the backup group, with g in $\{1, 2, ..., D\}$ and where D is the number of groups.

- U_g is the number of users in the group g.
- R_{ujkg} is equal to 1 if the attachment A_{uj} is stored on device k in group g and 0 otherwise.
- R_{ujg} is the number of already stored replicas of the attachment A_{uj} by users in the backup group g. If R_{ujg} is greater than R then R_{ujg} is equal to R.

The equation for a user's energy cost when the user replicates the email message to her own devices is

$$E_u = (\sum_{i=1}^{N} B_{ui} + \sum_{j=1}^{M} A_{uj}) \times (Erd + Eft \times (R-1) + \sum_{k=1}^{R-1} Erd_k) \tag{3.4}$$

The equation for the user's energy cost when the user replicates the email message within the backup group is

$$Es_{ug} = E_u - \sum_{j=1}^{M} A_{uj} \sum_{k=1}^{R-1} R_{ujkg} \times (Eft + Erd_k) \tag{3.5}$$

The user's energy saving is

$$S_{Ug} = 1 - \frac{Es_{ug}}{E_u} = \frac{\sum_{j=1}^{M} A_{uj} \sum_{k=1}^{R-1} R_{ujkg} \times (Eft + Erd_k)}{(\sum_{i=1}^{N} B_{ui} + \sum_{j=1}^{M} A_{uj}) \times (Erd + Eft \times (R-1) + \sum_{k=1}^{R-1} Erd_k)} \tag{3.6}$$

In the case when Erd_k is equal to Erd, the number of replicas R is equal to 3, R_{ujkg} equals 1 for *User1* device and equals 0 for *User2* device, we can see that the equations 3.6 and 3.3 are identical. Let us assume that $Erd_k = \alpha_k * Erd$ and that $Eft/Erd = \beta$. If we then divide both parts of the fraction by Erd, equation 3.6 can be expressed as

$$S_{ug} = \frac{\sum_{j=1}^{M} A_{uj} \sum_{k=1}^{R-1} R_{ujkg} \times (\beta + \alpha_k)}{(\sum_{i=1}^{N} B_{ui} + \sum_{j=1}^{M} A_{uj}) \times (1 + \beta \times (R-1) + \sum_{k=1}^{R-1} \alpha_k)} \tag{3.7}$$

Let us make one simplification of equation 3.7 and assume that α is the same for other devices. It is a reasonable assumption because we can still model the cases when other devices Erd is different from the hub Erd. Equation 3.7 is then expressed as

$$S_{ug} = \frac{\sum_{j=1}^{M} A_{uj} \times (R_{ujg} - 1) \times (\beta + \alpha)}{(\sum_{i=1}^{N} B_{ui} + \sum_{j=1}^{M} A_{uj}) \times (1 + (\beta + \alpha) \times (R-1))} \tag{3.8}$$

If we divide both parts of the fraction by $(\beta + \alpha)$ and assume that $c = 1/(\beta + \alpha)$, then the equation 3.8 can be expressed as

$$S_{ug} = \frac{\sum_{j=1}^{M} A_{uj} \times (R_{ujg} - 1)}{(\sum_{i=1}^{N} B_{ui} + \sum_{j=1}^{M} A_{uj}) \times (c + R - 1)} \tag{3.9}$$

We can again see that the equation 3.9 is identical to equation 3.3 in case when α and β are equal to 1, R is equal to 3, R_j is equal to 2, and there is one message $(B + A)$ that needs to be replicated.

We can express average energy cost saving in the backup group as

$$S_g = \sum_{u=1}^{U_g} \frac{\left(\frac{\sum_{j=1}^{M} A_{uj} \times (R_{ujg} - 1)}{(\sum_{i=1}^{N} B_{ui} + \sum_{j=1}^{M} A_{uj}) \times (c + R - 1)}\right)}{U_g} \tag{3.10}$$

and the average cost savings in all groups as

$$S = \sum_{g}^{D} \frac{\left(\sum_{u=1}^{U_g} \frac{\left(\frac{\sum_{j=1}^{M} A_{uj} \times (R_{ujg} - 1)}{(\sum_{i=1}^{N} B_{ui} + \sum_{j=1}^{M} A_{uj}) \times (c + R - 1)}\right)}{U_g}\right)}{G_g} \tag{3.11}$$

From equation 3.11 we can make several observations about energy cost savings expectations. First, savings is inversely proportional to the number of replicas; i.e., the more replicas that are created, the less savings are realized. Second, coefficient c expresses Erd efficiency of the hub device relative to other devices and Eft to Erd efficiency of the hub device. If the Erd of other devices is the same as the hub device and the Eft of the hub device is equal to its Erd, then c is equal to 0.5. A smaller value of c indicates higher cost of an Eft hub device relative to its Erd, or the higher cost of other device Erd relative to the hub Erd, or both. The smaller c is, the higher the savings are; i.e., the higher the energy consumption of the devices participating in the backup group, the higher the savings, though the contribution of c to the savings is relatively small. Finally, $(R_{uig} - 1)$ is the number of other users in the backup group g having the same attachment A_j as user u. Consequently, the higher the number of users is in the backup group, the higher the number of the shared attachments is in the group; i.e., the higher number of users that are in the backup group, the higher the savings might be. But because the cost savings is averaged over all users in the group it is also possible that cost savings goes down if a user with lower cost savings is added to the group. Section 3.7.2 evaluates the cost model expressed by equation 3.11 for different number of replicas, number of users in the group using the social graph to infer the groups and randomly generating the groups, and values of coefficient c.

3.7.2 Evaluation

The energy-cost savings model expressed by the equation 3.11 is evaluated on the private and Enron email datasets and the crawled Flickr dataset. Flickr does not provide the photo's size. It does include the photo's url in the photos info. We can use the *wget*[16] utility to query the photo size. But this approach is not scalable since there are over two million photos in the subgraph dataset. To work around this I retrieve the sizes for a random sample of images and then extrapolate the data to all images. The distribution of the sample's sizes is shown in Figure 3.15.

The cost model is evaluated for nine combinations of number of replicas, backup group sizes, and coefficient c. The number of replicas is in the range 2-10. The range is motivated by the availability analysis in Section 2.6 which concludes that a high level of availability can be achieved with 2-10 replicas. Backup groups are selected in several ways. First we

[16] `https://www.gnu.org/software/wget/manual/wget.html`

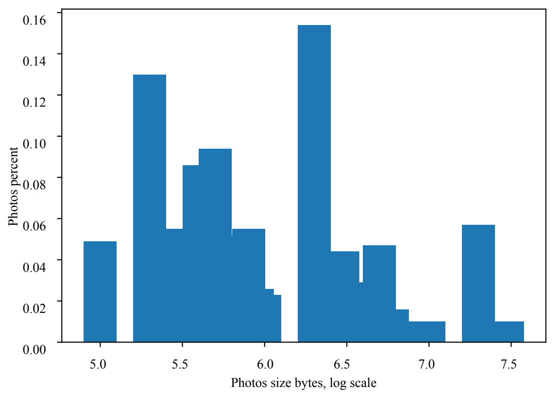
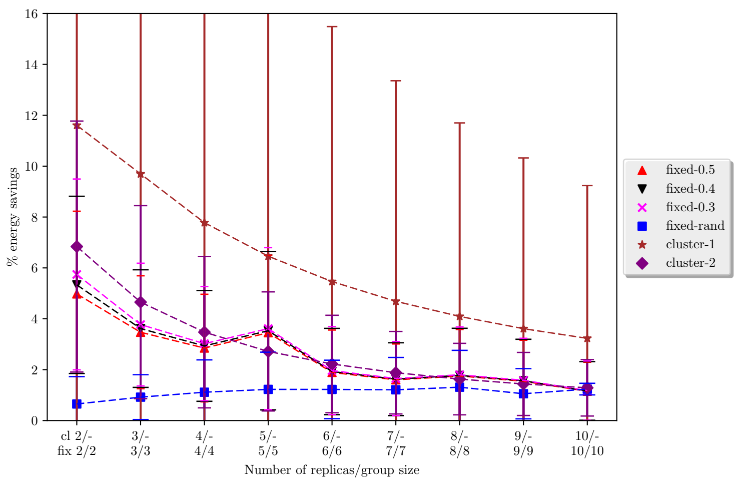


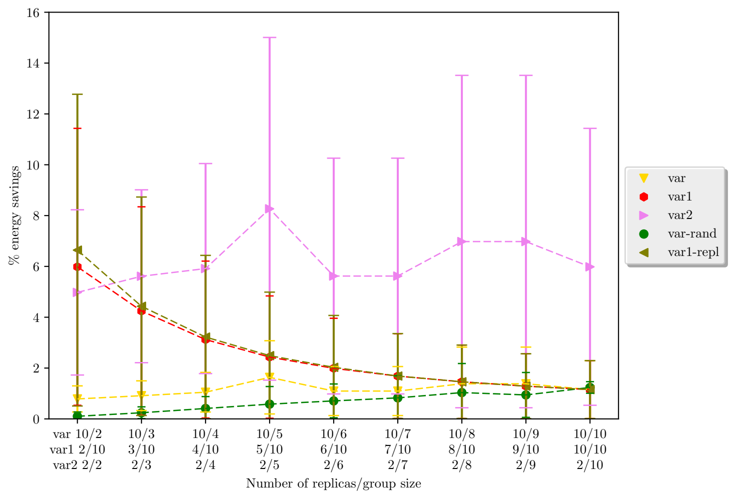
Figure 3.15: Distribution of photos size in the Flickr simple random sample of the photos.

use fixed group size, or groups with equal number of users. To create groups of equal size, for each user the nearest neighbor is found based on the weighted Jaccard similarity matrix where the weight is the size of the shared attachments between users. Then the symmetrical pairs of neighbors are chosen with the highest size of the shared attachments. Symmetrical pairs are those where if *user2* is the nearest neighbor of *user1* then *user1* is the nearest neighbor of *user2*. If there are not enough symmetrical pairs to have initial groups, where initial groups are calculated as the number of users in the network divided by the number of required replicas, then the remaining pairs are selected with the largest size of shared attachments. Note that pairs do not overlap; i.e., a user in one group can not appear in another group. If the number of replicas is greater than two, then the groups are extended by randomly choosing a user from the remaining users and adding it to a group with which the user has the maximum size of shared attachments. The motivation for a fixed group is the simplicity of the backup strategy where the number of replicas is the same as the number of users or devices in the backup group. That is if there are three replicas required then there are three devices to store these replicas. Having fixed group also gives a user a simple way to choose trusted users with whom to share the backup devices. The fixed group algorithm uses the social user relationship in group selection by choosing the nearest neighbor as the first member of the group. But the rest of the members are chosen so as to maximize attachment sharing within the group and without consideration of the social relationship between the users. Therefore, we can expect less than optimal cost saving in this approach. The second way of inferring the backup groups is by k-means clustering on the weighted Jaccard similarity matrix where the weight is the size of the shared attachments between users. This approach takes full advantage of social relationships between users, but depending on the social graph and the number of clusters, it may generate clusters or groups of varying size and high variation of energy-cost savings within the group. Finally, for comparison, a group of fixed size is generated by randomly choosing the group members. The set of coefficient c value's is $\{0.3, 0.4, 0.5\}$. If we assume that the Erd is the same for all devices in the backup group then the values correspond to Eft to Erd ratios of $\{2.3, 1.5, 1\}$. Unless specified otherwise, c is equal to 0.5. Overall, the following combination of replicas, groups, and coefficient c is evaluated:

- The number of replicas is increased from 2 to 10 with step size 1. The groups are fixed and the number of users in the group is equal to the number of replicas. Coefficient c is in the set $\{0.3, 0.4, 0.5\}$. This set of evaluation points is marked as



(a)



(b)

Figure 3.16: Energy cost savings evaluation in the private email archive. cluster-1 has 3 clusters, cluster-2 has 10 clusters.

$fixed - 0.5, fixed - 0.4, fixed - 0.3$ on the plot legend.

- This test is the same as above but the groups are randomly generated. This set of evaluation points is marked as $fixed - rand$ in Figures plot legend.

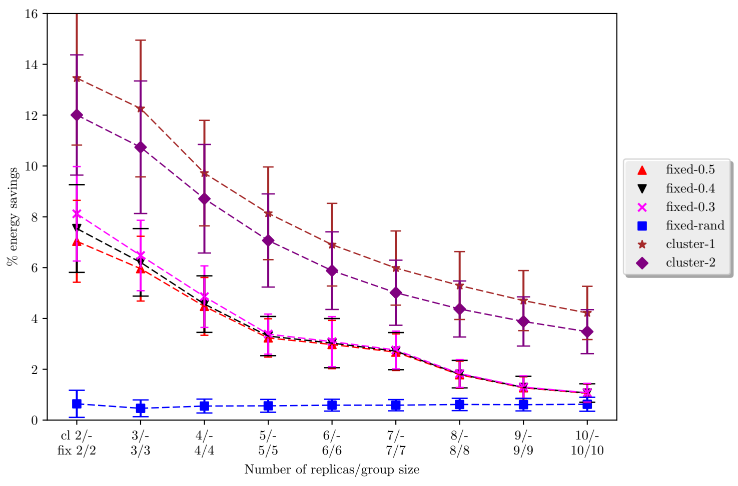
- The number of replicas is increased from 2 to 10 with step size 1. The groups are inferred by k-means clustering and are the same regardless of the number of replicas. The number of clusters depends on the dataset and each dataset is evaluated using two clusters. This set of evaluation points is marked as $cluster - 1, cluster - 2$ on the plot legend.

- The number of replicas is increased from 2 to 10 with step size 1. The number of users in the backup group is fixed at 10. In this case it is assumed that there is a strategy to choose users to store the replicas so that each user has the same number of replicas from different users. For instance, in the case of two replicas, five pairs of users can be chosen with the users within each pair storing each other replicas.

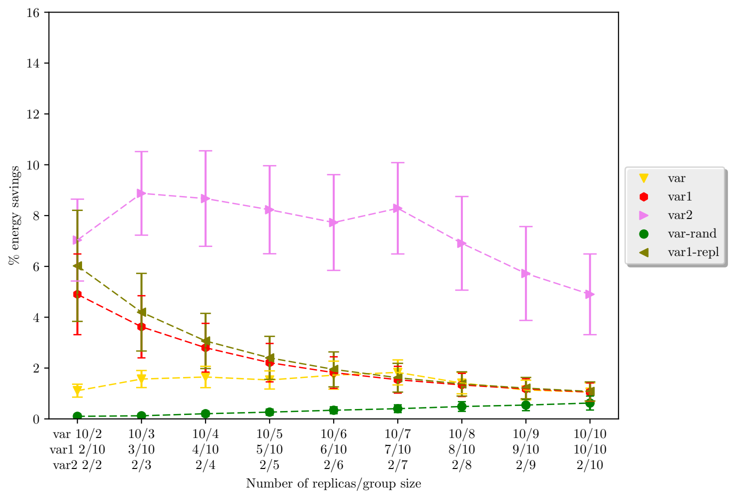
This strategy is outside of the scope of this research since it does not affect the cost model at the group level. The strategy is used whenever the number of replicas is less than the number of users in the backup group. This set of evaluation points is marked as $var1$ on the plot legend.

- This test is the same as above but the groups are randomly generated. This set of evaluation points is marked as $var1 - rand$ on the plot legend.
- The number of replicas is constant and is equal to 10. The number of users in the backup group is increased from 2 to 10 with step size 1. When the number of required replicas is greater than the number of the users or devices in the group then the difference in required replicas is provided by the user. For instance, the user provides an additional eight devices if there are 10 required replicas and only two users in the backup group. We can expect that cost savings in this case are small because there are fewer shared attachment between the users. This strategy is used whenever the number of replicas is greater than the number of users in the backup group. This set of evaluation points is marked as var on the plot legend.
- This test is the same as above but the number of replicas is equal to two. This set of evaluation points is marked as $var2$ on the plot legend.
- This test is the same as above but the groups are randomly generated. This set of evaluation points is marked as $var - rand$ on the plot legend.
- In all tests it is assumed that the hub device stores all of the user's attachments. This ensures that the user always has at least one complete replica of all emails. Within a backup group, an attachment can aggregately have more replicas than required. We can devise a strategy where the backup group aggregately stores exactly the required number of the attachment replicas. For instance, if the backup group has 10 users in it with six required replicas and there are already six replicas of the attachment A on six devices owned by different users and a new email is received with attachment A by a user from the backup group but not one of the users who already has this attachment, then attachment A is not stored on this user's device. This provides an additional energy cost savings at the expense of the user not owning the complete set of replicas of her email and some overhead is necessary to manage the replicas location. There is one test evaluating this strategy. It is a modification of test $var1$. This set of evaluation points is marked as $var1 - repl$ on the plot legend.

Evaluation of the tests above is shown in Figures 3.16 for the private email archive, 3.17 for the Enron archive, and 3.18 for the Flickr dataset. Evaluation of the private email archives and Enron dataset is split into two Figures for better visualization. Error bars on each plot show 95% confidence intervals. The dashed line connects average values for better visualization of the results. The x-axis tick labels have multiple rows of labels. The labels that apply to a specific test have a first tick label with a prefix that corresponds to the specific test. For instance, the tick labels that start with the prefix *cl* correspond to the cluster test. The x-axis ticks show the value of the required replicas and the number of users in the group. In the case of k-means inferred clusters, the number of users in the backup group varies but the number of the clusters is fixed. The Figure's title provides the details of the number of clusters in each dataset and specific test. The number of clusters



(a)



(b)

Figure 3.17: Energy cost savings evaluation in the Enron email dataset. cluster-1 has 10 clusters, cluster-2 has 15 clusters.

in the private email archive evaluation is set to 3 and 10, which roughly corresponds to each cluster having 10 and 3 users. The number of clusters in the Enron dataset is set to 10 and 15, which roughly corresponds to each cluster having 15 and 10 users. The number of clusters in the Flickr dataset is set to 82 and 246, which roughly corresponds to each cluster having 15 and 5 users. The main motivation for the chosen number of clusters and the maximum size of 10 for the users in the backup group is the idea of having backups on the devices of a closely related circle of friends and family. Consequently, I apply the two smallest clustering of relationships that tend to occur at 5 (support cliques) and 12-15 (sympathy group) users, as suggested by Hill and Dunbar in [77].

There are a number of observations that can be made from the plots in Figures 3.16, 3.17, and 3.18:

- In all cases, the randomly generated groups ($fixed-rand, var-rand$), regardless of other parameters, cost saving slightly increases as the number of users in the group increases but is less than 2%, is substantially smaller than most of other tests, and reaches about half the cost savings of most of other tests when the number of users in the group is equal to 6, 9, and 10 in the private, Enron and Flickr dataset, respectively. This validates the idea that social relationships can be used to optimize replica placement in backup strategy. A slight increase in cost savings is expected since the number of shared attachments uniformly increases as the number of the users in the backup group increases.

- As expected, we can see from the Figures 3.16a and 3.17a that the smaller coefficient c, that is the higher energy cost of Eft versus Erd in the hub device or the higher cost of other devices Erd versus the hub device Erd, slightly increases the cost savings $(fixed - 0.3, fixed - 0.4, fixed - 0.5)$. But cost savings are under 1% and are rapidly decreasing when increasing the number of replicas and users in the group.

- As expected, we can see from Figures 3.16b, 3.17b, and 3.18 $(fixed - *, cluster - *, var1*,)$ that cost savings decrease as the number of the replicas increases, regardless of the number of the users in the backup group. In fact, the power trend-line with a negative power in the range 0.5-0.7 fits the test points with R^2 in 0.89-0.98 range.

- As expected, we can see from Figures 3.16b and 3.17b $(var1 - repl)$ that fixing the total number of replicas slightly increases cost savings but the cost savings are at most 1% and rapidly decrease as the number of replicas increases.

- As expected, we can see from Figures 3.16b and 3.17b $(var2)$ that increasing the number of users in the group may increase cost savings. We see in the private email archive evaluation that the $var2$ test is increasing until the number of users in the backup group is nine and then it decreases. In the Enron email dataset there is an increase in cost savings when the number of users in the group increases from two to three and from six to seven but decreases afterwards.

- We see from the Figures 3.16a, 3.17a, and 3.18 that clustering has the greatest cost savings in all cases if we consider it relative to the increase in the number of replicas. These data further validate the idea of using social relationships to optimize the backup strategy.

- We see a high variation of values in all the tests except for the $var - rand$ and var tests in the private email archive in Figure 3.16. Variation is the highest in the $cluster - 1$ test with three clusters because the created clusters have high variance in average cost-saving within each cluster. The reason for the high variation is that the private email archive is a small dataset with 29 users, where the majority of users are naturally grouped into couples. We see much smaller variation in the Enron dataset, which has 151 users, and even smaller variation in the Flickr dataset, which has 1 231 users.

We can conclude that social relationships can be used to optimize the backup strategy in the decentralized email architecture by choosing the backup group with users who have more social contact with each other; i.e., in the context of email there are more shared email attachments. The cost of the saving in groups selected this way tends to decrease as the number of replicas and/or users in the backup group increases. If we choose the number of replicas to be six then with k-means clustering we can construct backup groups that have saving around 6%-15%. While savings in single digits may appear small, if we

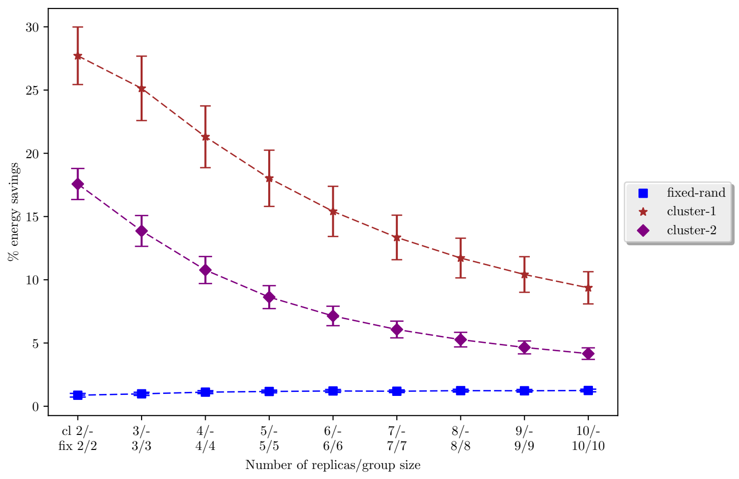


Figure 3.18: Energy cost savings evaluation in the crawled Flickr dataset. cluster-1 has 82 clusters, cluster-2 has 246 clusters.

consider it at the scale of data centers, then the numbers are significant. Indeed, 5% of 416.2 terawatt hours of electricity used by datacenters in 2015 is 20.8 terawatt, which is enough to supply electricity to New York City for about four months[17].

3.8 Limitations

Below I identify some limitations of my methodology and analysis:

- Use of email datasets in research is notoriously difficult because of privacy concerns. Moreover, privacy concerns are compounded by the fact that this research is looking at email attachments as the way of extracting social networks. Consequently this thesis is limited in terms of the analysis by two email datasets - private email archives from family and friends and the largest publicly available email dataset of the Enron email corpus. Private archives are collected from 29 users and are biased because it is the ego network of the thesis' author. Enron's dataset contains messages from 151 core users and has been extensively used in SNA and Natural Language Processing research. While the datasets are small, and our ability for generalization of conclusions is limited, we can see similarity between the two datasets in terms of the general email attachments statistics and the k-nearest neighbor and k-means clustering results. To some extent, this validates our approach to the extraction of the network and its analysis. Moreover, we corroborate our finding by using the same approach on the Flickr online photo-sharing social network.

- Email archives are a snapshot of the email messages at the time that the messages are downloaded. This means that we might be missing connections between users because of deleted messages. Moreover, users have different email habits and while some may keep all their emails, others generally delete most of them retaining only what is needed. Short of having an email logger on an email server, there

[17] `http://www.nyiso.com/public/webdocs/media_room/publications_presentations/Power_Trends/Power_Trends/2016-power-trends-FINAL-070516.pdf`

is no way to address this issue. As the result of this bias we can expect that the structure we discover in the network maybe biased as well. The extent of the bias is difficult to estimate. Interestingly, the structure discovered from the private archive matched perfectly with the actual social relationships of the group. In Enron, we stipulate that the discovered structure reflects the functional rather than organizational hierarchies and we do not have any available information for the former.

- In order to evaluate energy cost saving, we have to know the dataset sizes, and shared information, namely email attachments in the case of email. We can obtain this information from email archives, but it could be a limiting factor in other datasets. For instance, Flickr does not natively provide photos-size information. It does include the photos url in the photos information. This allows us to obtain the photo size by querying the url. But this approach might not be scalable when dealing with millions of photos. To work around this, we can obtain the images sizes of the random sample and then extrapolate this data to all images.

- Flickr only allows the ability to query for forward links. Consequently, it is not possible to crawl a complete, large, weakly-connected component. Moreover, due to the high computational demands required for social structure discovery, we analyzed a subgraph of nodes with degree greater than 30 000. This is a biased and densely connected core of users. In such dataset the groups might blend with each other. Nevertheless, we demonstrate that even in this dense group of users the discovered social structures are significant and distinct from random groups.

- The energy cost model does not take into consideration the initial cost of the backup or the backup bootstrapping. The issue in question is how to choose the backup group when a user creates a decentralized email account for the first time. The methodology for choosing the backup group depends on the availability of social data. The decision of how to choose the backup group can be postponed until this data becomes available. A user may have enough devices to support the required number of replicas in the meantime. But this may result in the loss of email data in case of a catastrophic failure. Assuming that the user already has an email account with one of the centralized services, email messages can be imported into the user's IoT devices. In fact, if the user already has an email account, then messages can be imported from one of the user's client devices and then social information can be extracted from these messages and from the messages of the user's circle of trusted family and friends can be analyzed to choose the backup group. Another option is to use other social networks of which the user is a member to infer the backup group. Dunbar et al. in [57] suggests that online social networks may mirror offline networks. Consequently, the social structure of online networks may mirror the social structure extracted from email archives and therefore be used to infer the backup group. The cost of the initial backup group selection can be estimated by increasing the required number of replicas by the number of the replicas that do not remain in the backup group. For instance, if there are six required replicas, and one additional replica is needed for the initial backup, then we can assume there are seven replicas in total. Note, however, that in practice, initial replicas are required for fewer messages and therefore the cost is not going to be as high as for storing complete replica of all messages.

3.9 Summary

I used Enron, private email, and Flickr OSN datasets in my analysis. I linked the EDRMV2 Enron email dataset containing attachments with the Shetty and Adibi Enron email dataset, which has been extensively used in previous research. I then extracted two social networks for each email dataset. The first has nodes representing users and ties representing communication between users. The second has nodes representing users and ties representing email attachments shared between users. I suggested a set of rules to filter out TRAM attachments like common logos, signatures, or Internet trend images, which are commonly shared between people and do not represent a valuable tie. Four centrality measures - degree, eigenvector, betweenness, and closeness were calculated and the resulting ranked list of users in the top 10 of each measure was analyzed for each network. I demonstrated that the k-nearest neighbor model accurately predicts similarity between users. In the Enron network, this is corroborated by an employee's position in the organizational chart. In the private network, the nearest neighbor is the socially closest person. K-means clustering in the Enron network shows groups with users in each group related to each other via their functional responsibilities. In the private network, groups represent core family cells like husband and wife or two siblings and close friends. Extracting the social network from shared attachments could be complementary to the generally used communication network and may discover more key influential people in the network and help to infer FOAF relationship. Moreover, I demonstrated how the analysis could be used to optimize the backup strategy in the proposed decentralized email architecture. In addition, I showed how the same type of analysis can be generalized to the Flickr online photo-sharing social network. I demonstrated that the groups constructed with k-means clustering based on favorites photos sharing correlate with the user's contacts. I further demonstrated how this analysis can be used to optimize placement of photo replicas on devices of socially related users to optimize the energy cost. This strategy could be used in a decentralized social network like Diaspora[18]. I see two main challenges in extracting the shared attachments network. First is defining the quality of ties, and second is defining the strength of the tie. Both of these topics are left for future research. In this chapter I demonstrated the viability of extracting social network from email shared attachments. Because of privacy concerns, it is extremely difficult to obtain an email corpus for the analysis. Consequently, my evaluation and ability to generalize is constrained by available data. I hope that this research will motivate large email providers to apply my approach in their SNA research.

[18]https://diasporafoundation.org

Chapter 4

Conclusions

4.1 Summary of thesis

Email has been available for over 50 years. It was the first killer application and, to date, is the most preferred way of communication. Its success can be attributed to its simplicity of use, reliability, availability, and free cost, with most of the benefits provided by centralized services. But there are intangible costs to users. A user's privacy is affected since the providers, in order to support the infrastructure and make a profit, sell user data to advertisers. While users of the centralized services offer great value by providing the "raw material" such as their emails, profiles, interests, and friends, they are increasingly disenfranchised from the online economy. Moreover, centralized services are high-value targets to hackers and subject to government surveillance. In addition, legacy email protocols defined over 20 years, have been adapting to the changing environment, such as the growth of mobile devices, by enhancing protocols with extensions and, consequently, making them more complex. The review of related work in Section 2.1.3 shows that current research based largely on the P2P overlay network does not address email issues. On the other hand, in recent years, a new phenomena of IoT has evolved, presenting opportunities to researchers and consumers. IoT is expected to generate a large amount of heterogeneous data, including personal data from smart homes, smart cars, or smart wearables. Researchers see the need for data analytics to take place at the edge, where the data are collected. This presents an opportunity for the consumer to become an active trader of her data and benefit from the digital economy. Consequently, the development of IoT presents an opportunity for the decentralization of online services, including email. The reason for this is the multitude of smart IoT devices owned by a user. Devices are continuously connected to the Internet and have some computing resources available.

In Section 2.1.4 I posit the research question of the feasibility of a decentralized email system based on resource-constrained devices, which is characteristic of the IoT environment. In Section 2.2 I propose a high-level email architecture with message revision maintenance and a synchronization protocol replacing the legacy IMAP protocol. I consider some typical email use cases that can be addressed by this architecture. The architecture is then evaluated in Section 2.4. Analysis shows that the architecture is feasible on a resource-constrained device like Raspberry Pi, and that it performs at least as well as the centralized email system in terms of disk and energy usage, latency, memory, and CPU. Importantly, it also shows that not every back-end that is capable of email revision control works. Git, a popular revision control system with content-addressed storage and Merkle-tree synchronization, is CPU, storage, and energy bound because of the way it maintains revision snapshots. Analysis also shows that synchronization of a single large file, for instance the dataset created by Sqlite database, by the *rsync* copy and synchronization utility can be energy bound because of the way it reassembles synchronized blocks on the destination device.

I show in Section 2.7 that data centers make a substantial impact on global energy demand and consequently on green house gas emissions. It is, therefore, an important question to ask how the proposed decentralized email architecture compares against the centralized email architecture. My analysis, based on Google data center metrics and the evaluation in Section 2.4, shows that the proposed architecture is an efficient solution in terms of energy consumption and could be as much as 3.4 times more efficient than a centralized architecture.

In Section 2.6 I discuss the availability of a decentralized email architecture. The analysis takes into consideration previous research into the availability of P2P email systems, P2P storage, a distributed file system, and decentralized social networks. The overarching conclusion is that, depending on the availability of individual devices, 2 to 10 content replicas are sufficient to provide availability comparable with the availability offered by centralized services. In the IoT environment, the user may own a number of devices providing abundant storage and connectivity. There may be enough devices to have an adequate number of content replicas. A user may tolerate intermittent availability, but the user has zero tolerance for data loss. IoT devices are a single point of failure because they generally are located at a user's home. Consequently, some content replicas have to be placed on the devices outside of a user's home providing both durability and high availability of the data. Therefore, it is important to make the backup strategy part of the decentralized architecture. It is only natural to consider the user's social circle of family and friends as the providers of their own IoT devices to store additional replicas or backups. The benefit is clearly mutual in this case. The reasons are that there is a high level of trust enjoyed by the small clique of people and that they tend to interact with each other more, meaning they already share some of their data such as emails, favorite pictures, blogs or articles. However, the question I ask is how to choose the circle of friends composing the backup group. I suggest that understanding the social relationships of users provides the answer to this question. Chapter 3 explores this suggestion in detail.

As part of my research, in Table 3.1 I analyze email statistics for the Enron email corpus and the private email archives of family and friends. Statistics show that 81-91% of email message space is used by email attachments and 27-34% of email message space is used by duplicate email attachments. Moreover, 25-29% of the attachments are shared between users. The proposed architecture handles duplicate attachments in an efficient way by providing content-addressed storage and Merkle-tree based synchronization. As suggested above, further optimization can be accomplished through the backup strategy and the high number of shared attachments gives weight to this idea. Email is a classic collaborative application. As such, information exchange is the goal of a user's interactions. Since attachments take up most of the email's space, in Section 3.1.4 I posit the research question of what can be learned from SNA of the network extracted from email's attachment interactions and how it can be applied to the task of optimization.

In Section 3.2 I cover network extraction via email attachments shared between users. This approach is used in SNA analysis of OSN but is novel in the context of the email archive as a related work review demonstrates in Section 3.1.3.1. Tie strength definition is an important part of network extraction as shown in Section 3.1.3.2. Tie definition in shared attachments network presents a challenge because multiple attachments in the same message can represent user interactions over multiple (for instance, pictures from two different parties) or single (for instance, pictures from one party) events. Another challenge is identifying what I collectively call TRAM attachments: company logos, e-signatures, and trend Internet images. I analyze the pattern of the spread of attachments in the Enron email corpus and private email archives. Based on this analysis, I propose a set of rules to eliminate TRAM attachments.

Depending on research goals, there are different approaches to SNA, as the review of related work shows in Section 3.1.3.3. I chose degree, betweenness, closeness, and eigenvector centrality measures to analyze the shared attachments network and compare it to a conventionally-used communication network. In addition, I use the commonly used algorithms of k-nearest neighbor and k-means clustering. Overall, the analysis in Sections 3.3 and 3.5 demonstrates that both networks are complementary to each other and either one or both can be used in SNA depending on the research question. Centrality measures show the most influential users in terms of communication and information exchange. K-nearest neighbor classification results are similar for both networks in the Enron email corpus, while in private email archive communication network prediction is not accurate. K-means clustering in the case of the communication network in the Enron email corpus produces clusters better centered around departments than in the attachments network. Clustering for the communication network in the private email archive, just like in k-nearest neighbor, is not accurate.

In Section 3.6 I demonstrate that the proposed way of network extraction and analysis can be extended to OSN. I crawled the photo-sharing Flickr social network to extract a dataset with 2 000 177 nodes. To reduce the computational load required for structural analysis of such a large network, I extract a subgraph of nodes with the degree higher than 30 000. The resulting network has 1 231 nodes and 631 115 unique edges and is a densely connected core. I show that the clusters of users extracted from this network by using the k-means clustering algorithm form a social structure and their similarity is statistically more significant than uniformly randomly generated clusters.

In Section 3.7 I define the energy cost-savings model from the backup strategy based on shared email attachments. The model is evaluated on the private email archive, Enron data, and Flickr datasets. Combination of the parameters of the group size, number of replicas, the energy efficiency coefficient, and the way that the backup group is inferred, is used as the input into the model to evaluate the energy cost savings from various backup strategies. I hypothesize that social relationships between users can be used to optimize the backup strategy. The k-nearest neighbor algorithm and k-means clustering are used on the email attachments Jaccard similarity matrix to build groups of various sizes. I demonstrate that the backup groups extracted with this approach have higher energy cost savings than the randomly generated groups. Moreover, I show that in the photo-sharing Flickr social network, placing photo replicas on socially related peers results in significant cost savings relative to randomly selected peers. This demonstrates that the proposed analysis can be extended beyond a decentralized email architecture to other decentralized architectures based on socially connected users where it could be used for architectural decisions requiring optimal content replication on peers.

4.2 Future work

Research presented in this thesis is work in progress. The main focus of this thesis is on evaluation of a resource-constrained IoT device, like Raspberry Pi, to support basic email functions of the decentralized email architecture and the application of the SNA to optimize the replication strategy of email messages by minimizing the replication's energy cost. Consequently, I only provide a high level decentralized email architecture, sufficient to demonstrate its viability. Decentralized email architecture has many parts to it and is a large project to cover in one thesis. Therefore, there are a number of limitations of this research, some of which were identified in Sections 2.5 and 3.8. Other limitations are subject to future research. I attempted to start the discussion on how a decentralized email architecture might take advantage of IoT phenomena and what tools can be used for its optimization. Consequently, there are many directions for a future research, of which I present only a subset.

Message format needs to be reviewed. While keeping MIME body parts is important for optimal storage and partial synchronization, some of the headers may no longer be applicable in an IoT environment where email relaying is not necessary.

Back-end storage requires a detailed architecture. I outlined some desired back-end features like content addressed storage and Merkle tree based synchronization to optimize an attachment's storage and synchronization between devices. I also showed that the Git approach to revision maintenance is not efficient. Consequently, additional evaluation is required of existing RCS implementations to find a solution with the features described above and optimal in terms of revision maintenance storage and access.

Synchronization with Merkle tree has to be enhanced using a partial download. This is an existing feature in IMAP server and is important for mobile devices.

Recipient public address discovery is no longer available from the DNS server in the decentralized architecture. I assume that at least one of the recipient devices has a globally accessible address but the sender is not aware of it. Discovery service can be delivered by existing email providers. Under this schema a user, just like today, has an account with a provider and registers her publicly accessible address(es) with it. The address(es) can change and have an expiration date. The sender, just like today, gets the provider address from DNS, and then contacts the provider to obtain the recipient's address.

Edge weight is set to 1 in the shared attachments network. Should it be higher if there are multiple attachments in the email message? Do attachments in the same message but from different events, for instance images from two parties or unrelated documents, affect the weight? These questions require further evaluation of existing data.

TRAM (e-signature, company logo, trendy image, etc.) attachments may significantly affect the extracted network. In an extreme case, it could make every user in the network connected. For instance, a company's logo automatically may be included in every email within an organization. In this thesis, based on empirical evidence, I suggested a set of rules to exclude this type of attachment. But the rules are not comprehensive and the evaluation of the rules was limited to visual validation when possible. The problem is inherently difficult for two reasons. First, there is a lack of data for research. To my knowledge, the Enron email corpus is the largest publicly available email dataset. Moreover, the vast majority of attachments in this dataset are work related; i.e., documents, and not images as one would expect in a private email archive. Second, TRAM attachments, while not too dissimilar from SPAM in some cases are, by and large, indistinguishable from "normal" attachments, and therefore difficult to identify. One path for research

could be evaluation of unsupervised ML algorithms to classify attachments as TRAM.
Feature vector selection in k-nearest neighbor and k-means clustering in communication network shows significantly different results in terms of classification and clustering accuracy in the Enron email corpus and private email archive. Further analysis and evaluation is needed to identify the appropriate feature vector.
General applicability of the shared attachment network to existing research is a substantial area open to investigation. The question is whether we can improve current SNA of a communication network by applying additional knowledge obtained from the analysis of the shared attachments network. For instance, can we make better predictions of supervisor-subordinate relationship or organization hierarchy?

Bibliography

[1] AOL. https://en.wikipedia.org/wiki/AOL.

[2] ARPANET. https://en.wikipedia.org/wiki/ARPANET.

[3] Compatible Time Sharing System. https://en.wikipedia.org/wiki/Compatible_Time-Sharing_System.

[4] Enron Scandal. https://en.wikipedia.org/wiki/Enron_scandal.

[5] Official Biography: Raymond Tomlinson. http://internethalloffame.org/official-biography-raymond-tomlinson.

[6] Webmail. https://en.wikipedia.org/wiki/Webmail.

[7] Apache Wave. https://en.wikipedia.org/wiki/Apache_Wave, 2010.

[8] Gmail data loss affects 150,000 users. http://blog.backup-technology.com/5844/gmail-data-loss-affects-15000-users/, 2011.

[9] Google's Green Computing: Efficiency at Scale. https://static.googleusercontent.com/media/www.google.com/en//green/pdfs/google-green-computing.pdf, 2011.

[10] Email Client Popularity. https://www.campaignmonitor.com/resources/will-it-work/email-clients/, 2012.

[11] Edward Snowden: Leaks that exposed US spy programme. http://www.bbc.com/news/world-us-canada-23123964, 2014.

[12] Cisco Global Cloud Index: Forecast and Methodology, 20152020. http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf, 2016.

[13] Microsoft announces largest wind energy purchase to date. https://news.microsoft.com/2016/11/14/microsoft-announces-largest-wind-energy-purchase-to-date/#So45wxOAtMElu25Y.97, 2016.

[14] Online activities ranked by regular usage penetration in Great Britain as of March 2016. https://www.statista.com/statistics/289048/online-activities-ranked-by-usage-penetration-great-britain-uk/, 2016.

[15] Hillary Clinton email controversy. https://en.wikipedia.org/wiki/Hillary_Clinton_email_controversy, 2017.

[16] Social network analysis. https://en.wikipedia.org/wiki/Social_network_analysis, 2017.

[17] Apoorv Agarwal, Adinoyi Omuya, Aaron Harnly, and Owen Rambow. A Comprehensive Gold Standard for the Enron Organizational Hierarchy. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, number July, pages 161–165, Jeju, Republic of Korea, 2012.

[18] Luiz Alves, Matheus Maciel, Lesandro Ponciano, and Andrey Brito. Assessing the Impact of the Social Network on Marking Photos as Favorites in Flickr. In *WebMedia'12*, number iii, pages 79–82, 2012.

[19] William Lintner Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo. United States Data Center Energy Usage Report. (September), 2016.

[20] Eric Auchard. Exclusive: Big data breaches found at major email services - expert. http://www.reuters.com/article/us-cyber-passwords-idUSKCN0XV1I6, 2016.

[21] James E. Bartlett, Joe W. Kotrlik, and Chadwick C Higgins. Organizational Research: Determining Appropriate Sample Size in Survey Research Appropriate Sample Size in Survey Research. *Information Technology, Learning and Performance Journal*, 19(1):43–50, 2001.

[22] Tom Bawden. Global warming: Data centres to consume three times as much energy in next decade, experts warn. http://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html, 2016.

[23] Sivan Bercovici, Yaniv Frishman, Idit Keidar, and Ayellet Tal. Decentralized electronic mail. In *Proceedings - International Conference on Distributed Computing Systems*, 2006.

[24] Hal Berghel. Email The Good , The Bad , and The Ugly. *Communications of the ACM*, 40(4):11–15, 1997.

[25] J.D. Biersdorfer and David Pogue. *The Internet. The missing manual.* O'Reilly Media Inc, 2006.

[26] Mohd Khairun Nasir Bin Sa'adi. *An Offline Email Implementation Based on Delay Tolerant Network.* PhD thesis, 2013.

[27] Christian Bird, Alex Gourley, Prem Devanbu, and Michael Gertz. Mining Email Social Networks. In *MSR'06*, pages 137–143, Shanghai, China, 2006.

[28] Andrew D. Birrell, Roy Levin, Michael D. Schroeder, and Roger M. Needham. Grapevine: an exercise in distributed computing. *Communications of the ACM*, 25(4):260–274, 1982.

[29] William J. Bolosky, John R. Douceur, David Ely, and Marvin Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. *Performance Evaluation Review*, 28(1):34–43, 2000.

[30] Stephen P. Borgatti. Identifying sets of key players in a social network. *Computational and Mathematical Organization Theory*, 12(1):21–34, 2006.

[31] Pj Braam. The Coda distributed file system. *Linux Journal*, pages 46–50, 1998.

[32] Brandon Gaille. Time Spent Online Statistics by Region and Type of Activity. http://brandongaille.com/time-spent-online-statistics-by-region-and-type-activity/, 2013.

[33] Aurelien Breeden, Sewell Chan, and Nicole Perlroth. Macron Campaign Says It Was Target of Massive' Hacking Attack. https://www.nytimes.com/2017/05/05/world/europe/france-macron-hacking.html?_r=0, may 2017.

[34] Eric Brewer. Towards Robust Distributed Systems. In *ACM Symp.Principles of Distributed Computing (PODC 00) (invited talk)*, pages 7–10, Portland, Oregon, 2000. ACM.

[35] Eric Brewer. CAP twelve years later: How the "rules" have changed. *Computer*, 45(2):23–29, 2012.

[36] Dotan Di Castro, Zohar Karnin, Liane Lewin-Eytan, and Yoelle Maarek. You've got Mail, and Here is What you Could do With It! Analyzing and Predicting Actions on Email Messages. In *WSDM '16 Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 307–316, 2016.

[37] V Cerf, S Burleigh, A Hooke, L Torgerson, R Durst, K Scott, K Fall, and H Weiss. RFC: 4838. *Network Working Group*, pages 1–36, 2007.

[38] Meeyoung Cha, Alan Mislove, Ben Adams, and Krishna P. Gummadi. Characterizing social cascades in Flickr. In *Proceedings of the first workshop on Online social networks - WOSP '08*, pages 13–18, 2008.

[39] Dave Chaffey. Mobile Marketing Statistics Compilation. http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/, 2016.

[40] Moon-Ok Choi, Il-Woo Lee, and Ho-Jin Park. An Asynchronous Peer-to-Peer Service Extension Using the General E-mail Service. In *2008 10th International Conference on Advanced Communication Technology*, pages 1739–1741. Ieee, feb 2008.

[41] Munmun De Choudhury, Winter A Mason, Jake M Hofman, and Duncan J Watts. Inferring Relevant Social Networks from Interpersonal Communication. In *WWW 2010*, pages 301–310, Raleigh, North Carolina, USA, 2010.

[42] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J J Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. Spanner : Google's Globally-Distributed Database. In *Proceedings of OSDI'12: Tenth Symposium on Operating System Design and Implementation*, pages 251–264, 2012.

[43] Andy Crabtree, Tom Lodge, James Colley, Chris Greenghalgh, and Richard Mortier. Accountable IoT? Outline of the Databox model. In *International Symposium on a World of Wireless, Mobile, and Multimedia Networks*, pages 1–6, 2017.

[44] Andy Crabtree, Tom Lodge, James Colley, Chris Greenhalgh, Richard Mortier, and Hamed Haddadi. Enabling the new economic actor: data protection, the digital economy, and the Databox. *Personal and Ubiquitous Computing*, 20(6):947–957, 2016.

[45] Laura Creekmore. In Praise of Email. *Information Architecture*, 43(1):25–27, 2016.

[46] M Crispin. RFC 3501: Internet Message Access Protocol - version 4rev1. *Network Working Group*, pages 1–109, 2003.

[47] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook : a Privacy Preserving Online Social Network Leveraging on Real-Life Trust. *Communications Magazine, IEEE*, 47(12):94–101, 2009.

[48] Ajit D. Dhiwal, Sudip Gautam, Akshay K. Singh, and Vijay K. Chaurasiya. Mail-Zoro - Email based P2P file sharing. In *Proceedings of the 2008 16th International Conference on Networks, ICON 2008*, 2008.

[49] Christopher P Diehl, Galileo Namata, and Lise Getoor. Relationship Identification for Social Network Discovery. In *The AAAI 2008 Workshop on Enhanced Messaging, AAAI Conference On Artificial Intelligence*, pages 546–552, 2008.

[50] Jana Diesner and Kathleen M Carley. Exploration of Communication Networks from the Enron Email Corpus. In *Proc. of the Workshop on Link Analysis, Counterterrorism and Security, SIAM Intl. Conf. on Data Mining*, pages 3–14, 2005.

[51] Jana Diesner, L. Terrill Frantz, and M. Kathleen Carley. Communication Networks from the Enron Email Corpus It ' s Always About the People . Enron is no Different . *Computational & Mathematical Organization Theory*, 11(3):201–228, 2006.

[52] John R Douceur. The Sybil Attack. `https://www.microsoft.com/en-us/research/wp-content/uploads/2002/01/IPTPS2002.pdf`, 2002.

[53] John R Douceur, Atul Adya, William J Bolosky, Dan Simon, Marvin Theimer, and P Simon. Reclaiming space from duplicate files in a serverless distributed file system. In *ICDCS 2002: Proceedings of the 22nd International Conference on Distributed Computing Systems*, number July, pages 617–624, 2002.

[54] Robin I.M. Dunbar. Co-Evolution of Neocortex Size, Group Size and Language in Humans. *Behavioral and Brain Sciences*, 16:1–27, 1993.

[55] Robin I.M. Dunbar. The Social Brain Hypothesis. *Evolutionary Antropology*, pages 178–190, 1998.

[56] Robin I.M. Dunbar. How Many Friends Can You Really Have? `http://spectrum.ieee.org/telecom/internet/how-many-friends-can-you-really-have`, 2011.

[57] Robin I.M. Dunbar, Valerio Arnaboldi, Marco Conti, and Andrea Passarella. The structure of online social networks mirrors those in the offline world. *Social Networks*, 43:39–47, 2015.

[58] D. 3rd Easlake, Huawei, and T. Nansen. RFC 6234: US Secure Hash Algorithms. *Internet Engineering Task Force*, 2011.

[59] W K Edwards, E D Mynatt, K Petersen, M J Spreitzer, D B Terry, and M M Theimer. Designing and Implementing Asynchonous Applications with Bayou. In *User Interface Softw. and Techno. (UIST)*, pages 119–128, Banff, Alberta, Canada, 1997.

[60] Michael H Fischer, T J Purtell, and Monica S Lam. Email Clients as Decentralized Social Apps in Mr. Privacy. In *HotPETs 2011: Hot Topics in Privacy Enhancing Technologies*, pages 1–10, 2011.

[61] N Freed and N Borenstein. RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. *Network Working Group*, (November), 1996.

[62] M Gagnaire, F Diaz, C Coti, and C Cerin. Downtime statistics of current cloud solutions. *The International Working Group on Cloud Computing Resiliency*, (March):2–4, 2012.

[63] Thomas Gazagnaire, Amir Chaudhry, Jon Crowcroft, Anil Madhavapeddy, Richard Mortier, David Scott, David Sheets, and Gregory Tsipenyuk. Irminsule ; a branch-consistent distributed library database. pages 2–4, 2014.

[64] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. In *SOSP'03*, volume 37, page 29, Bolton Landing, New York, USA, 2003.

[65] Eric Gilbert and Karrie Karahalios. Predicting tie strength with social media. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 211–220, 2009.

[66] Vindu Goel and Nicole Perlroth. Yahoo Says 1 Billion User Accounts Were Hacked. `https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html`, dec 2016.

[67] Mark S. Granovetter. The Strength of Weak Ties. *American Journal of Sociology*, 78(6 (May 1973)):1360–1380, 1973.

[68] Mihajlo Grbovic, Guy Halawi, Zohar Karnin, and Yoelle Maarek. How Many Folders Do You Really Need?: Classifying Email into a Handful of Categories. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 869–878, 2014.

[69] Jean-Loup Guillaume and Matthieu Latapy. Bipartite graphs as models of complex networks. *Physica A: Statistical Mechanics and its Applications*, 371(2):795–813, 2006.

[70] A Gulbrandsen. RFC 4978: The IMAP COMPRESS Extenstion. *Network Working Group*, 2007.

[71] Mangesh Gupte and Tina Eliassi-Rad. Measuring tie strength in implicit social networks. In *Proceedings of the 3rd Annual ACM Web Science Conference on - WebSci '12*, pages 109–118, 2012.

[72] Mangesh Gupte, Pravin Shankar, Jing Li, S. Muthukrishnan, and Liviu Iftode. Finding hierarchy in directed online social networks. In *Proceedings of the 20th international conference on World wide web - WWW '11*, page 557, 2011.

[73] Hamed Haddadi, Heidi Howard, Amir Chaudhry, Jon Crowcroft, Anil Madhavapeddy, and Richard Mortier. Personal Data: Thinking Inside the Box. pages 29–32, 2015.

[74] Sufian Hameed, Muhammad Arsal Asif, and Farhan Kamal Khan. PiMail : Affordable , Lightweight and Energy-Efficient Private Email Infrastructure. In *International Conference on Innovations in Information Technology*, pages 296–301, 2015.

[75] J S Hardin and P C Urc. Network Analysis with the Enron Email Corpus. *Journal of Statistics Education*, 23(2), 2015.

[76] Luke Harding. Top Democrat's emails hacked by Russia after aide made typo, investigation finds. `https://www.theguardian.com/us-news/2016/dec/14/dnc-hillary-clinton-emails-hacked-russia-aide-typo-investigation-finds`, dec 2016.

[77] R. A. Hill and Robin I.M. Dunbar. Social network size in humans. *Human Nature*, 14(1):53–72, 2003.

[78] Pan Hui, Jon Crowcroft, and Eiko Yoneki. BUBBLE rap: Social-Based Forwarding in Delay Tolerant Networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 241–250, 2008.

[79] Emir Husni and Agus Wibowo. E-mail system for delay tolerant network. In *Proceedings of the 2012 International Conference on System Engineering and Technology, ICSET 2012*, 2012.

[80] Mohammad Jaber, Peter T Wood, Panagiotis Papapetrou, and Sven Helmer. Inferring offline hierarchical ties from online social networks. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 1261–1266, Seoul, Korea, 2014.

[81] Jaron Lanier. *Who Owns The Future.* Simon & Schuster, New York, New York, USA, 2013.

[82] David Jensen and Jennifer Neville. Data Mining in Social Networks. *Dynamic Social Network Modeling and Analysis workshop summary and papers*, pages 7–9, 2002.

[83] Edson Kageyama, Carlos Maziero, and Altair Olivo Santin. An Experimental Peer-to-Peer E-mail System. In *2008 11th IEEE International Conference on Computational Science and Engineering*, pages 203–208. Ieee, jul 2008.

[84] J. Kangasharju, K.W. Ross, and D.a. Turner. Secure and resilient peer-to-peer e-mail design and implementation. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, pages 184–191. IEEE Comput. Soc, 2003.

[85] T Kaye, D Khatami, D Metz, and E Proulx. Quantifying and comparing centrality measures for network individuals as applied to the Enron corpus. `https://www.siam.org/students/siuro/vol7/S01320.pdf`, 2014.

[86] Przemysław Kazienko, Radosław Michalski, and Sebastian Palus. Social network analysis as a tool for improving enterprise architecture. In *Proceedings of the 5th International KES Symposium on Agents and Multi-agent Systems, KES-AMSTA 2011.*, volume 6682, pages 651–660, Manchester, UK, 2011.

[87] J Klensin. RFC 2821: Simple Mail Transfer Protocol. *Network Working Group*, (April), 2001.

[88] David Koll, Jun Li, and Xiaoming Fu. SOUP: An Online Social Network By The People, For The People. In *Middleware'14*, pages 143–144, Bordeaux, France, 2014.

[89] Joshoua Kopstein. The Mission to Decentralize the Internet. `http://www.newyorker.com/tech/elements/the-mission-to-decentralize-the-internet`, 2013.

[90] Ravi Kumar, Novak Jasmine, and Andrew Tomkins. Structure and Evolution of Online Social Networks,. In *KDD'06*, Philadelphia, Pennsylvania, USA, 2006.

[91] Michal Laclavík, Štefan Dlugolinský, Marcel Kvassay, and Ladislav Hluchý. Email social network extraction and search. In *Proceedings - 2011 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT 2011*, volume 3, pages 373–376, 2011.

[92] Michal Laclavík and Martin Šeleng. Emails as Graph : Relation Discovery in Email Archive. In *Www 2012 - Companion*, pages 841–846, Lyon, France, 2012.

[93] H.-Y. Lam and D.-Y. Yeung. A Learning approach to spam detection based on social networks. In *4th conference on Email and anti-spam ({CEAS 2007})*, pages 81–89, 2007.

[94] B Leiba. RFC 2177: IMAP4 IDLE command. *Network Working Group*, pages 1–4, 1997.

[95] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[96] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A Survey and Comparison of Peer-To-Peer Overlay Network Schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005.

[97] Mary Madden. Public Perception of Privacy and Security in the Post-Snowden Era. `http://www.pewinternet.org/2014/11/12/public-privacy-perceptions/#`, 2014.

[98] Marcelo Maia, Jussara Almeida, and Virgilio Almeida. Identifying user behavior in online social networks. In *Proceedings of the 1st workshop on Social network systems - SocialNets '08*, pages 1–6, 2008.

[99] Peter V Marsden and Karen E Campbell. Measuring Tie Strength. *Social Forces*, 63(2):482–501, 1984.

[100] Andrew McCallum, Xuerui Wang, and Andrés Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. *Journal of Artificial Intelligence Research*, 30:249–272, 2007.

[101] Matt McGee. Email Is Top Activity On Smartphones, Ahead Of Web Browsing & Facebook. `http://marketingland.com/smartphone-activities-study-email-web-facebook-37954`, 2013.

[102] Miller Mcpherson, Lynn Smith-lovin, and James M Cook. Birds of a Feather : Homophily in Social Networks. *Annual Review of Sociology*, 27:415–444, 2001.

[103] B.Yin M.Duczynski. Hierarchy Detection through Email Network Analysis. pages 109–117, 2006.

[104] A Melnikov, D Cridland, and C Wilson. RFC 5162: IMAP4 Extensions for Quick Mailbox Resynchronization. *Network Working Group*, pages 1–24, 2008.

[105] Ralph C Merkle. A Digital Signature Based on a Conventional Encryption Function. http://www.springerlink.com/index/q865hwxq73ex1am9.pdf, 1988.

[106] Patrik Emanuel Mezo, Mircea Vladutiu, and Lucian Prodan. Distributed Mailing System. In *2011 IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pages 349–354, 2011.

[107] Patrik Emanuel Mezo, Mircea Vladutiu, and Lucian Prodan. HMail: A Hybrid Mailing System Based on the Collaboration between Traditional and Peer-to-Peer Mailing Architectures. In *2012 7th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 255–260, Timisoara, Romania, 2012. IEEE.

[108] Radoslaw Michalski and Przemyslaw Kazienko. Matching Organizational Structure and Social Network Extracted from Email Communication. *Lecture Notes in Business Information Processing*, 87:290–300, 2011.

[109] Alan Mislove, Ansley Post, Andreas Haeberlen, and Peter Druschel. Experiences in building and operating ePOST, a reliable peer-to-peer application. In *EuroSys'06*, volume 40, page 147, Leuven, Belgium, oct 2006.

[110] Alan Mislove, Ansley Post, Charles Reis, Paul Willmann, Peter Druschel, Dan S Wallach, Xavier Bonnaire, Pierre Sens, Jean-michel Busca, and Luciana Arantes-bezerra. POST : A Secure , Resilient , Cooperative Messaging System. In *HOTOS'03 Proceedings of the 9th conference on Hot Topics in Operating Systems*, Berkeley, USA, 2003.

[111] G Namata, Lise Getoor, and Christopher P Diehl. Inferring formal titles in organizational email archives. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[112] Pam Neely. 8 Reasons Why Email is Still the Killer App For Now. `https://www.act-on.com/blog/8-reasons-why-email-is-still-the-killer-app-for-now/`, 2016.

[113] L Nerenberg. RFC 3516: Binary Content Extension. *Network Working Group*, 2003.

[114] C Newman. RFC 5248: A Registry for SMTP Enhanced Mail System Status Codes. *Network Working Group*, 2006.

[115] Craig Partridge. The Technical Development of Internet Email. *IEEE Annals of the History of Computing*, 30:3–29, 2008.

[116] Nishith Pathak and Jaideep Srivastava. Automatic Extraction of Concealed Relations from Email Logs (Extended Abstract) . In *NetSci2006*, pages 2–4, Bloomington, IN, USA, 2006.

[117] Charith Perera, Susan Wakenshaw, Tim Baarslag, Hamed Haddadi, Arosha Bandar, Richard Mortier, Andy Crabtree, Irene Ng, Derek McAuley, and Jon Crowcroft. Valorizing the IoT Databox: Creating Value for Everyone. *Transactions on Emerging Telecommunication Technologies*, 2016.

[118] Raluca Ada Popa, Emily Stark, Steven Valdez, Jonas Helfer, Nickolai Zeldovich, and Hari Balakrishnan. Building Web Applications on Top of Encrypted Data Using Mylar. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 157–172, 2014.

[119] J A Pouwelse, P Garbacki, J Wang, A Bakker, J Yang, A Iosup, D H J Epema, M Reinders, and M R Van Steen. TRIBLER : a social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, (May 2007):127–138, 2008.

[120] Sara Radicati. Email Statistics Report, 2015-2019. `http://www.radicati.com/wp/wp-content/uploads/2015/02/Email-Statistics-Report-2015-2019-Executive-Summary.pdf`, 2015.

[121] Sara Radicati. Instant Messaging Statistics Report, 2015-2019. `http://www.radicati.com/wp/wp-content/uploads/2015/03/Instant-Messaging-Statistics-Report-2015-2019-Executive-Summary.pdf`, 2015.

[122] A Rae, B Sigurbjörnsson, and R van Zwol. Improving tag recommendation using social networks. *Adaptivity, Personalization and Fusion of Heterogeneous Information*, pages 92–99, 2010.

[123] B Y Rodrigo Rodrigues and Peter Druschel. Peer-to-Peer Systems. *Communications of ACM*, 1999.

[124] J. Rosenberg, C. F. Everhart, and N. S. Borenstein. An overview of the Andrew message system. *ACM SIGCOMM Computer Communication Review*, 17(5):99–108, 1987.

[125] Charalampos Rotsos, Heidi Howard, David Sheets, Richard Mortier, Anil Madhavapeddy, Amir Chaudhry, and Jon Crowcroft. Lost In the Edge : Finding Your Way With Signposts. In *3rd USENIX Workshop on Free and Open Communications on the Internet*, 2013.

[126] R Rowe, G Creamer, S Hershkop, and S J Stolfo. Automated social hierarchy detection through email network analysis. In *Joint 9th WebKDD and 1st SNA-KDD Workshop 2007 on Web Mining and Social Network Analysis. Held in conjunction with 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2007*, pages 109–117, 2007.

[127] Yasushi Saito, Eric Hoffman, Brian Bershad, Henry Levy, D. Becker, and B. Folliot. The Porcupine scalable mail server. In *Proceedings of the 8th ACM SIGOPS European workshop on Support for composing distributed applications*, volume M, page 52, 1998.

[128] Nishanth Sastry and Jon Crowcroft. SpinThrift : Saving Energy in Viral Workloads. In *Green Networking*, pages 69–75, 2010.

[129] Nishanth Sastry, Eiko Yoneki, and Jon Crowcroft. Buzztraq : Predicting geographical access patterns of social cascades using social networks. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, 2009.

[130] M Satyanarayanan. A Survey of Distributed File Systems. *Annual Review of Computer Science*, 4(4976):73–104, 1990.

[131] Chris Schmandt and Stefan Marti. Active Messenger : E-Mail Filtering and Delivery in a Heterogeneous Network. *Human-Computer Interaction*, 20:163–194, 2005.

[132] Tushar Semwal and Shivashankar Nair. AgPi: Agents on Raspberry Pi. *Electronics*, 5(4):72, 2016.

[133] Rajesh Sharma, Anwitaman Datta, Matteo Dell'Amico, and Pietro Michiardi. An empirical study of availability in f2f storage systems. In *IEEE International Conference on Peer-to-Peer Computing*, volume 1, pages 348–351, 2011.

[134] Jitesh Shetty and J Adibi. Discovering Important Nodes through Graph Entropy: The Case of Enron Email Database. In *Proceedings of the 3rd International Workshop on Link Discovery*, pages 74–81, Chicago, Illinois, 2005.

[135] Jitesh Shetty and Jafar Adibi. The Enron Email Dataset - Database Schema and Brief Statistical Report. *Information Sciences Institute Technical Report,*, 2004.

[136] Gyuwon Song, Suhyun Kim, and Daeil Seo. Replica placement algorithm for highly available peer-to-peer storage systems. In *1st International Conference on Advances in P2P Systems, AP2PS 2009*, pages 160–167, 2009.

[137] Jagan Srinivasan, Wei Wei, Xiaosong Ma, and Ting Yu. EMFS: Email-based personal cloud storage. In *Proceedings - 6th IEEE International Conference on Networking, Architecture, and Storage, NAS 2011*, pages 248–257, 2011.

[138] Jon Stokes. Google Wave : why we didn't use it. `http://arstechnica.com/information-technology/2010/08/google-wave-why-we-didnt-use-it/`, 2010.

[139] John Tang, Cecilia Mascolo, Dipartimento Fisica, and Vito Latora. Temporal Distance Metrics for Social Network Analysis. In *WOSN'09*, pages 31–36, Barcelona, Spain, 2009.

[140] John Tang, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Vincenzo Nicosia. Analysing Information Flows and Key Mediators through Temporal Centrality Metrics. In *Proceedings of the 3rd Workshop on Social Network Systems*, 2010.

[141] Lei Tang, Geoffrey Barbier, Huan Liu, and Jianping Zhang. A social network analysis approach to detecting suspicious online financial activities. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6007 LNCS, pages 390–397. Springer-Verlag Berlin Heidelberg, 2010.

[142] Kanchana Thilakarathna and H Petander. Enabling mobile distributed social networking on smartphones. In *MSWiM'12*, pages 357–366, Paphos, Cyprus, 2012.

[143] Kanchana Thilakarathna, Aline Carneiro Viana, Aruna Seneviratne, and Henrik Petander. Design and Analysis of an Efficient Friend-to-Friend Content Dissemination System. *IEEE Transactions on Mobile Computing*, 16(3):702–715, 2017.

[144] D N Tran, F Chiang, and J Li. Friendstore: cooperative online backup using trusted nodes. In *Proceedings of the 1st Workshop on Social Network Systems*, pages 37–42, 2008.

[145] Joshua R. Tyler, Dennis M. Wilkinson, and Bernardo A. Huberman. Email as Spectroscopy: Automated Discovery of Community Structure within Organizations. *The Information Society*, 21(2):143–153, 2003.

[146] Masoud Valafar, Reza Rejaie, and Walter Willinger. Beyond friendship graphs: a study of user interactions in Flickr. In *WOSN'09*, pages 25–30, Barcelona, Spain, 2009.

[147] Adam Vaughan. Google uses AI to cut data centre energy use by 15%. `https://www.theguardian.com/environment/2016/jul/20/google-ai-cut-data-centre-energy-use-15-per-cent`, 2016.

[148] Tom Van Vleck. The history of electronic mail. `http://www.multicians.org/thvv/mail-history.html`.

[149] J.R. von Behren, S. Czerwinski, A.D. Joseph, E.a. Brewer, and J. Kubiatowicz. NinjaMail: the design of a high-performance clustered, distributed e-mail system. In *International Workshop on Parallel Processing*, pages 151–158. IEEE Comput. Soc, 2000.

[150] Feng Wang, Kuai Xu, and Haiyan Wang. Discovering shared interests in online social networks. In *Proceedings - 32nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW 2012*, pages 163–168, 2012.

[151] Min Feng Wang, Sie Long Jheng, Meng Feng Tsai, and Cheng Hsien Tang. Enterprise email classification based on social network features. In *Proceedings - 2011 International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2011*, pages 532–536, 2011.

[152] Steve Whittaker, Tara Matthews, Julian Cerruti, Hernan Badenes, and John Tang. Am I wasting my time organizing email ? A study of email refinding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*, volume 13, pages 3449–3458, 2011.

[153] Zhe Wu and Changjia Chen. User classification and relationship detecting on social network site. In *2011 International Conference on Control, Automation and Systems Engineering, CASE 2011*, pages 2–5, 2011.

[154] Ibrar Yaqoob, Ejaz Ahmed, Ibrahim Abaker Targio Hashem, Abdelmuttlib Ibrahim Abdalla Ahmed, Abdullah Gani, Muhammad Imran, and Mohsen Guizani. Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges. *IEEE Wireless Communications*, 24(3):10–16, 2017.

[155] K Yelupula and Srini Ramaswamy. Social network analysis for email classification. In *Proceedings of the 46th Annual Southeast Regional Conference on XX - ACM-SE 46*, page 469, 2008.

[156] Cheng Yong, Wu Jiangjiang, Mei Songzhu, Wang Zhiying, Jun Ma, Ren Jiangchun, and Yan Ke. Mailbook: Privacy-Protecting Social Networking via Email. In *Proceedings of the Third International Conference on Internet Multimedia Computing and Service - ICIMCS '11*, page 90, 2011.

[157] Lejun Zhang, Tongxin Zhou, Qi Zhixin, Lin Guo, and Li Xu. The Research on E-mail Users' Behavior of Participating in Subjects Based on Social Network Analysis. *China Communications*, 13(4):70–80, 2016.

[158] Jianxin Zhao, Richard Mortier, Hamed Haddadi, and Jon Crowcroft. Towards Security in Distributed Home System. In *Proceedings of the 12th EuroSys Conference*, Belgrade, 2017. ACM.

[159] Yue Zhao, Shuigeng Zhou, and Aoying Zhou. E-Mail Services on Hybrid P2P Networks. *Lecture notes in Computer Science*, (60373019):610–617, 2004.

[160] Ding Zhou, Yang Song, and Hongyuan Zha. Towards Discovering Organizational Structure from Email Corpus. In *Proceedings of the Fourth International Conference on Machine Learning and Applications (ICMLA05)*, 2005.

[161] Yingjie Zhou, Kenneth R. Fleischmann, and William A. Wallace. Automatic text analysis of values in the enron email dataset: Clustering a social network using the value patterns of actors. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 1–10, 2010.

[162] Yingjie Zhou, Mark Goldberg, M. Magdon-Ismail, and W.A. Wallace. Strategies for cleaning organizational emails with an application to enron email dataset. In *5th Conf. of North American Association for Computational Social and Organizational Science*, number 0621303, Emory - Atlanta, Georgia, 2007.

[163] Mark Zuckerberg. New Year Resolution. https://www.facebook.com/zuck/posts/101043801707145
2018.

Appendix A

Enron dataset processing

In my analysis I use the Shetty and Adibi [135] dataset linked to the EDRM Version Two Enron corpus dataset. Initially, William Cohen from CMU prepared the Enron dataset and published it for researchers[1]. The dataset contains 517 424[2] emails from 151 users with 242 944 unique email content SHA1. Shetty and Adibi removed duplicate messages from the dataset and fixed some email address discrepancies. Their dataset contains 252 759 email messages with 212 326 unique email content SHA1. To verify that Shetty and Adibi is a subset of the Cohen dataset I first checked that *Message-ID* set from the Cohen dataset is a superset of the Shetty and Adibi dataset. Second, I collected SRS of 384 (Bartlett et al. [21]) emails from the Shetty and Adibi dataset and verified via email content SHA1 that emails match those in the Cohen dataset. As part of the verification I had to make some fixes in the Shetty and Adibi email message body: 1) 14 messages had X-FileName header included in the content; 2) One message had its body truncated. Neither the Cohen nor the Shetty and Adibi dataset contain attachments. EDRM released two versions of Enron dataset with attachments. Version One (EDRMV1) contains 697 079 emails with 155 431 unique email content SHA1 for 130 users. Version Two (EDRMV2) contains 1 234 387 emails with 242 800 unique email content SHA1 for 151 users. The EDRMV1 dataset does not provide continuity and applicability of my analysis to previous body of research because data are missing for 21 employees. I therefore decided to use the EDRMV2 dataset as the main source of attachments in my analysis. However, I discovered that email addresses in *From, To, Cc*, and *Bcc* header fields in the EDRMV2 dataset do not conform to canonical email address format of *user@domain* and are most likely taken from the original Enron email corpus. Here are some instances of Phillip Allen email addresses:

"ALLEN PHILLIP K" <pallen@enron.com>
"phillip.k.allen" <phillip.k.allen@enron.com>
<Allen>, "Phillip"
<Allen>, "Phillip K."
<Phillip.Allen@enron.com>
<Allen>, "Phillip K." </O=ENRON/OU=NA/CN=RECIPIENTS/CN=Pallen>
Phillip K Allen <Phillip.K.Allen@enron.com>
Allen, Phillip K.

[1] `http://www-2.cs.cmu.edu/~enron`

[2] This is seven less emails than the originally released dataset of 517 431 emails. Seven emails were removed by William Cohen for privacy reasons.

Phillip K Allen
Phillip,K,Allen

Those formats are partially consistent with what was found by Zhou et al. in [162]. However, the problem is compounded by email lists having comma separated addresses in mixed format. Here are some examples:

Brad,Alford,Phillip,K,Allen,anderson,Bob
Kristin Albrecht, Phillip K Allen, Hunter S Shively
Frolov, Yevgeny </O=ENRON/OU=NA/CN=RECIPIENTS/CN=Yfrolov>, Allen, Phillip K.

Parsing of these lists is not trivial and error prone. I therefore decided to take the EDRMV2 dataset and replace the *From, To, Cc*, and *Bcc* header fields with ones from the Shetty and Adibi dataset. While the email address in the Shetty and Adibi dataset is in canonical form of name@domain, there are multiple email addresses used by a single Enron employee. For instance, Phillip Allen has six email addresses:

k..allen@enron.com
phillip.k.allen@enron.com
pallen@ect.enron.com
pallen@enron.com
phillip.allen@enron.com
philip.allen@enron.com

To identify a set of email addresses used by all core Enron employees I created a dataset of all addresses in the *From, To, Cc, Bcc* header fields which match one of the Enron's core employee last name. There are 3 282 email addresses in this dataset. I then manually reviewed this list to identify core employees addresses. The resulting list contains 492 email addresses.

I link emails from the EDRMV2 and Shetty and Adibi datasets via employee, folder, subject, date, and email content SHA1 key. I cannot link on *Message-ID* because the EDRMV2 dataset has its own generated *Message-ID*. This key uniquely identifies 252 722 emails in Shetty and Adibi and 754 906 emails in EDRMV2 datasets. However, the *Date* header field in the Shetty and Adibi dataset does not include Time Zone information and I discovered that the EDRMV2 dataset has, in some instances, the *Date* changed to a different Time Zone, consequently *Date* without the Time Zone will not match in otherwise identical emails in Shetty and Adibi and EDRMV2 datasets. My solution is to link the Cohen dataset to EDRMV2 dataset with dates converted to GMT and filter each email by *Message-ID* from the Shetty and Adibi dataset. For simplicity, I refer in the text below to the Shetty and Adibi dataset rather than Cohen's filtered by Shetty and Adibi dataset.

I had to do the following data processing in order to achieve the best linking of the Shetty and Adibi dataset to the EDRMV2 dataset:

- I calculate SHA1 in both datasets by extracting all email content, removing all new lines and spaces, mapping all quotable-printable[3] characters to '?' and then

[3]`https://en.wikipedia.org/wiki/Quoted-printable`

replacing all multiple occurrences of '?' to a single '?'. This has to be done because the EDRMV2 dataset has email content reformatted with many instances of non-ASCII characters. To get the SHA1 for the EDRMV2 dataset consistent with the SHA1 in the Shetty and Adibi dataset, I remove *boundary*[4] lines from EDRMV2 email content and remove all content following and including a copyright notice inserted by EDRM.

- Some content in the EDRMV2 dataset is truncated. I matched 1 990 emails from the Cohen dataset by truncating emails to the same length as in the EDRMV2 dataset. I set minimum length of email content in those cases to 100 bytes to provide for sufficient entropy in the text.

- I had to make changes to match dates between two datasets. In the EDRMV2 dataset, when the email folder is "schedule_crawler", the time is 4 hours behind the corresponding Shetty and Adibi time. In addition, I found the following pattern where EDRMV2 time is behind Shetty and Adibi time by the number of hours specified in a zone difference with Coordinated Universal Time (UTC) or by 2, 3, 4, 10 and 12 hours. Overall 1 225 linked emails had the date fixed in this manner.

- For 8 627 messages I had to "downgrade" the key to subject, date, and SHA1.

Overall 249 353 out of 252 754 emails were linked, which represents 98.6% of emails from Shetty and Adibi dataset; 3 401 emails were not linked. Missing emails are distributed over 125 employees. Further analysis shows that the top two, Jeff Dasovich (5.11%) and Richard Sanders (50.85%), from this list account for 55.96% of the missing emails. Judging by SHA1 analysis, missing employees from the Shetty and Adibi dataset do not have corresponding emails in the EDRMV2 dataset. By linking missing emails to the EDRMV1 dataset via employee and SHA1 key, I recovered 1 579 an additional emails. This brings the total linked records to 99.28%. Missing emails are distributed over 119 employees with Jeff Dasovich and Richard Sanders accounting for 9.52% and 24.47% of missing emails, respectively.

[4] `https://tools.ietf.org/html/rfc2046`