12-2014

SDSF : social-networking trust based distributed data storage and co-operative information fusion.

Phani Chakravarthy Polina
University of Louisville

SDSF: SOCIAL-NETWORKING TRUST BASED DISTRIBUTED DATA STORAGE AND CO-OPERATIVE INFORMATION FUSION

By

Phani Chakravarthy Polina
M.S., Western Kentucky University, 2006

A Dissertation
Submitted to the Faculty of the
J.B.Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

December 2014

SDSF: SOCIAL-NETWORKING TRUST BASED DISTRIBUTED DATA STORAGE AND CO-OPERATIVE INFORMATION FUSION

By

Phani Chakravarthy Polina
M.S., Western Kentucky University, 2006

A Dissertation Approved On

September 25, 2014

by the following Dissertation Committee:

DEDICATION

Dedicated to my parents Mr. Polina Gopala Krishna and Mrs. Polina Baby, my wife Vahini Kotapati and my daughter Anshu Polina.

ACKNOWLEDGEMENTS

Taking up Ph.D. has been a great experience and I am grateful to all those who made it possible. First and foremost, I express my special appreciation and thanks to Dr. Anup Kumar, my advisor and mentor for every bit of his guidance, encouragement and inspiration. His valuable feedback, advice and expertise helped me tremendously with my research. His guidance is invaluable in writing this thesis without which, my Ph.D. would not have been achievable. He has provided very insightful discussions and feedback about my research.

I am also deeply thankful to Dr. Bin Xie for providing me an opportunity to work closely with him which immensely helped me to watch and learn from his work and experience. His support helped me feel confident to overcome the difficulties I have encountered. It is his supervisory role that helped me with continuing the research work for my Ph.D. study.

I express my sincere gratitude to Dr. Sunderesh Heragu for his confidence in me to offer the GRA position at LODI. He has been an excellent source of inspiration and support. I would also like to thank Dr. Rammohan Ragade for his continuous help since the start of my Ph.D. program. He was the first person to believe in me and helped me in getting my first job at UOFL. My thanks goes out to Dr. Darjen Chang and Dr. Adel S. Elmaghraby for agreeing to be members on my committee and for providing their valuable feedback.

A special thanks to Dr. Anala Pandit who has been an excellent support and a caring friend. I have also enjoyed all those useful and stimulating discussions. I would like to thank Dr Tuan T. Tran for his insightful feedbacks, discussions and constant encouragement. My appreciation goes to Dr. Shoeb Khan for his friendship.

I would also like to thank my parents for always being there for me and helping me in whatever way they could. I am also very grateful to my elder sister Femina Polina for her financial support throughout my studies. I would also like to thank my eldest sister Laveena Polina for her excellent inspiration. I would also like to thank Aruna kotapati for all her help with my daughter.

And to Vahini, my wife, who has been by my side, every day throughout my Ph.D., the faith she has in me kept me going through all my ups and downs.

ABSTRACT

SDSF: SOCIAL-NETWORKING TRUST BASED DISTRIBUTED DATA STORAGE AND CO-OPERATIVE INFORMATION FUSION

Phani Chakravarthy Polina

September 22, 2014

As of 2014, about 2.5 quintillion bytes of data are created each day, and 90% of the data in the world was created in the last two years alone. The storage of this data can be on external hard drives, on unused space in peer-to-peer (P2P) networks or using the more currently popular approach of storing in the Cloud. When the users store their data in the Cloud, the entire data is exposed to the administrators of the services who can view and possibly misuse the data. With the growing popularity and usage of Cloud storage services like Google Drive, Dropbox etc., the concerns of privacy and security are increasing. Searching for content or documents, from this distributed stored data, given the rate of data generation, is a big challenge. Information fusion is used to extract information based on the query of the user, and combine the data and learn useful information. This problem is challenging if the data sources are distributed and heterogeneous in nature where the trustworthiness of the documents may be varied.

This thesis proposes two innovative solutions to resolve both of these problems. Firstly, to remedy the situation of security and privacy of stored data, we propose an innovative Social-based Distributed Data Storage and Trust based co-operative Information Fusion Framework (SDSF). The main objective is to create a framework that assists in providing a secure storage system while not overloading a single system using a P2P like approach. This framework allows the users to share storage resources among friends and acquaintances without compromising the security or privacy and enjoying all the benefits that the Cloud storage offers. The system fragments the data and encodes it to securely store it on the unused storage capacity of the data owner's friends' resources. The system thus gives a centralized control to the user over the selection of peers to store the data.

Secondly, to retrieve the stored distributed data, the proposed system performs the fusion also from distributed sources. The technique uses several algorithms to ensure the correctness of the query that is used to retrieve and combine the data to improve the information fusion accuracy and efficiency for combining the heterogeneous, distributed and massive data on the Cloud for time critical operations. We demonstrate that the retrieved documents are genuine when the trust scores are also used while retrieving the data sources. The thesis makes several research contributions. First, we implement Social Storage using erasure coding. Erasure coding fragments the data, encodes it, and through introduction of redundancy resolves issues resulting from devices failures. Second, we exploit the inherent concept of trust that is embedded in social networks to determine the nodes and build a secure network where the fragmented data should be stored since the social network consists of a network of friends, family and acquaintances. The trust between the friends, and availability of the devices allows the user to make an informed choice about where the information should be stored using 'k' optimal paths. Thirdly, for the purpose of retrieval of this distributed stored data, we propose information fusion on distributed data using a combination of Enhanced N-grams (to ensure correctness of the query), Semantic Machine Learning (to extract the documents based on the context and not just bag of words and also considering the trust score) and Map Reduce (NSM) Algorithms.

Lastly we evaluate the performance of distributed storage of SDSF using erasure coding and identify the social storage providers based on trust and evaluate their trustworthiness. We also evaluate the performance of our information fusion algorithms in distributed storage systems. Thus, the system using SDSF framework, implements the beneficial features of P2P networks and Cloud storage while avoiding the pitfalls of these systems. The multi-layered encrypting ensures that all other users, including the system administrators cannot decode the stored data. The application of NSM algorithm improves the effectiveness of fusion since large number of genuine documents are retrieved for fusion.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Data storage and Information Fusion has been extensively studied in both academic and industrial domains. However, data security, scalability, and reliability are still not well solved although many products and technologies are proposed for data storage. This is due to a large number of data users and a huge amount of data generated every day [1]. Information Fusion is not well implemented when data is distributed and stored on multiple devices. How to effectively build scalable storage systems to reliably store, search and protect all the data for various users is extremely challenging. Localized storage such as mobile devices and hard drives is mostly used in our daily life. For data confidentiality, the data could be encrypted before writing them into the storage media. As each device is used separately (e.g., our laptops and our mobile phones), the localized data storage is subjected to a single-point of failure. In addition, separately using storage space of the user devices is inefficient as a large amount of storage space could be free on some devices while some devices may run out of storage space.

A Data Storage:

Network-based data storage has been rapidly developed in the past decades. It targets DDS systems and allows users to access data anywhere from the network. Notably, IBM Almaden Research Center at San Jose [2] initiated the research by proposing a system which prevents data failure by providing a certain degree of data redundancy at multiple places. The designed system offers much higher data resiliency and security, compared to the localized data storage. However, the data could be lost, illegally accessed, and stolen once the storage network (e.g.. a file database system) is attacked externally. In addition, scalability is another issue when the data size and number of users increase.

The network-based storage system recently evolved into peer-to-peer (P2P) storage, cloud storage, and an integration of both. The P2P systems like Ocenastore [3], Freenet [4], Dht [5], TotalRecall [6] utilize the storage space of the peers to store the data. When a data file is written into the P2P network, it is published to everyone in the community. The data files are organized in the network with the goal to maximize the device utilization. The file's creator (i.e., data owner) is unable to control and manage his file. In other words, P2P currently provides no mechanisms for users to effectively manage their data. The manipulations on the data are therefore very limited, e.g., the file may not be actually deleted, resulting in many security and privacy vulnerabilities. Furthermore, the availability of the stored information in P2P-based system highly depends on the users' interests (i.e., how many users want to download and store the file). The P2P-based storage systems can only provide a best-effort service without any performance guarantee. Due to the dynamics of the P2P-based networks (the users join and leave randomly), in many scenarios, the users cannot retrieve the desired information when wanted.

On the other hand, the cloud storage systems such as Dropbox [7] and Google Drive [8] store users' data into a pool of powerful servers. These systems enable users to reliably store their data files in an online drive. However, there exist several issues on data security and privacy because the users' data is open to the service provider. In addition, maintaining the data reliability is complex and expensive, especially when the system expands. More importantly, the storage capacity is limited and it is expensive to acquire more storage space. The users have to pay additional cost to upload and download the data to and from the systems. The cloud storage becomes an issue for storing large media files such as movies or archived data.

Towards addressing these challenges, we propose a Social-based Distributed Data Storage (SDSF) system for reliable and secure data storage and information fusion in a scalable and cost effective way. The storage nodes are effectively selected based on the social relationships with the data owner. To find the potential storage nodes, we provide an efficient algorithm to compute the social ties in the social networks. In addition, robust and efficient data encoding and decoding schemes are also used for data regeneration when a storage node fails.

There are other challenges in building the SDSF system:

- Erasure coding requires more computation and less storage space compared to replication. The erasure coding efficiency should be considered as the mobile storage nodes (e.g., mobile phones and tablets) are cheap devices and have limited computational capabilities. The erasure coding parameters such as (n, k) should be well determined to achieve the trade off between the available amount of storage space, the number of storage nodes, and the data resiliency. How to determine the potential storage nodes from a set of storage capable nodes is an important issue since each storage node has different capabilities like storage space, computational power, bandwidth and availability. The main problem while selecting storage nodes is how to select nodes that are trust worthy.

- Searching for information in huge data is by itself challenging. The problem becomes more difficult when the data is distributed among several machines. We search intelligently for data in real time.

- In a distributed storage system, parts of the file are stored in different storage devices. We have to ensure that even owners of the individual storage devices cannot read or modify the contents. The distributed storage system needs security to provide strong support for authentication, encryption and decryption. It has to authenticate a storage device and the user of the file.

- Faults (e.g., communication error, device failure and network disconnection) occur for many reasons. The mobile devices and tablets are not as stable as the fixed storage devices. Therefore, fault-handling is necessary to deal with the possible errors in the network or storage devices. The distributed storage protocol should have the capability to recover the system to a normal state from any failures.

Our SDSF makes the following contributions:

1. We design a system architecture for DDS based on the social networks. The proposed system is scalable to a large number of users.

2. We perform information fusion on the cloud using N-gram, Semantic Machine Learning, and Map Reduce (NSM) Algorithms. The innovation of the proposed NSM is to improve the information fusion accuracy and efficiency for processing heterogeneous, distributed, and massive data on the cloud for tactical information fusion services. The information fusion could start with a query request issued by a user, e.g., from a mobile device. The query consists of various fusion parameters, like keywords, geographic information, time and other contextual information, and they will be used for further filtering and fusion. Once the query is submitted to the fusion engine, NSM then performs information fusion as shown in Figure 19 to find relevant information.

3. We further provide a new framework for efficient data encoding and decoding based on the erasure coding for data reliability and security. In addition, we propose (N, K) coding schemes to efficiently process large data files.

4. For data security, we utilize the strong security properties of the Advanced Encryption Standard (AES) to encrypt the data. The data is encrypted with multiple layers ensuring that all other users, including system administrators, learn nothing about the stored data.

5. We propose an efficient algorithm to find the potential storage nodes based on the interactions in the social networks. The social ties are quantified and computed. This allows unused storage space to be utilized.

6. The performance of the proposed system is evaluated via both theoretical analysis and extensive simulations. In particular we have implemented a test bed which performs the experiments on real networks.

B Information Fusion

Before we look into the issues associated with Information Fusion and how to handle them, let us first look at Information Fusion, how it can be classified, and existing applications where Information/data fusion has been used. Historically, the word "Information Fusion" has been commonly reported to be used in the context of detection and classification tasks in a variety of application domains such as military applications and robotics [9]. In the recent times this term has been used in the context of intrusion detection [10], denial of services detection [11], improving location estimates of sensor nodes in wireless sensor networks, detecting routing failures in WSN [12], for intelligent traffic management [13] and also to collect link statistics for routing protocols [14]. Information Fusion has also been used extensively in military domain for autonomous weaponry employing multiple sensors; broad area surveillance systems employing single weapons platforms (e.g., ships, airborne surveillance, ground sites, surveillance spacecraft) or distributed sensor networks; fire control systems employing multiple sensors for acquisition, tracking, and command guidance; intelligence collection systems; indications and warning (I&W) systems, the mission of which is to assess threat and hostile intent and command for classification of underwater mines [15]; and control nodes for military forces. It is also used in robotics, machine intelligence, automated manufacturing, remote sensing, image processing, and medical systems. There exist many definitions of information fusion in fusion community. The first reported definition of data fusion has been by the data fusion work group of the Joint Directors of Laboratories (JDL) of US DoD as“multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from multiple sources” [16]. This definition was further refined to expand the scope by Hall and Llinias [17] as the combination of data from multiple sensors, and related information provided by associated databases, to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone [17].

However, the definition accepted by the International Society for Information Fusion in 2004 is “in the context of its usage in the society, it encompasses the theory, techniques and tools created and applied to exploit the synergy in the information acquired from multiple sources (sensor, databases, information gathered by humans, etc.) in such a way that the resulting decision or action is in some sense better (qualitatively or quantitatively, in terms of accuracy, robustness, etc.) than would be possible if any of these sources were used individually without such synergy exploitation” [18]. We use the term Information Fusion in this context.

All of the definitions in the literature agree that fusion is a process that is used to achieve some goals by combining data or information from multiple sources into one coherent format, in spite of their minor differences. Thus, the underlying reason why we need fusion is to achieve that goal of obtaining comprehensive information from multiple sources.

Information Fusion can be classified in multiple ways and one of the ways suggested by Nakamura et al [19] for classification of Information Fusion is based on the relationship between the sources. The classes are based on how much and what information does each source provide:

- Complementary Fusion:

  When a complete piece of information is obtained from sources that contain different portions of the information. i.e. Source 1 returns information portion ‘a’ and Source 2 returns information portion ‘b’ to create complete information ‘a+b’ using fusion. For example, temperature information from East, West, North, South parts of the monitored area can be fused to give the overall temperature of the monitored area.

- Redundant Fusion:
  If more than one source returns the same information, the accuracy of the information can be assumed to be better with higher confidence. This will typically result in fused information with more accuracy.

- Cooperative Fusion:
  This type of Information Fusion uses information from two independent sources and fuses them to obtain new information that is more complex than the information from each of the sources. For example, cooperative fusion can be used to obtain the dimensional images from two or more cameras at slightly different angles which are then combined to derive the three dimensional images of the observed scene.

Figure 1 shows the various types of information fusion based on the above classifications.

During the retrieval of information, the information has to be collected from various nodes. We need to collate this information from various sources. However, very little work has been done towards Information fusion for data stored in the cloud environment. There are several issues associated with “Information Fusion” for data in cloud environments:

- The information comes from various heterogeneous devices and may have different trust values associated with them. One or more sources in the system may not be trustworthy due to various reasons like bias, malice or faults in the source system. If these sources are unrecognized, false data may propagate

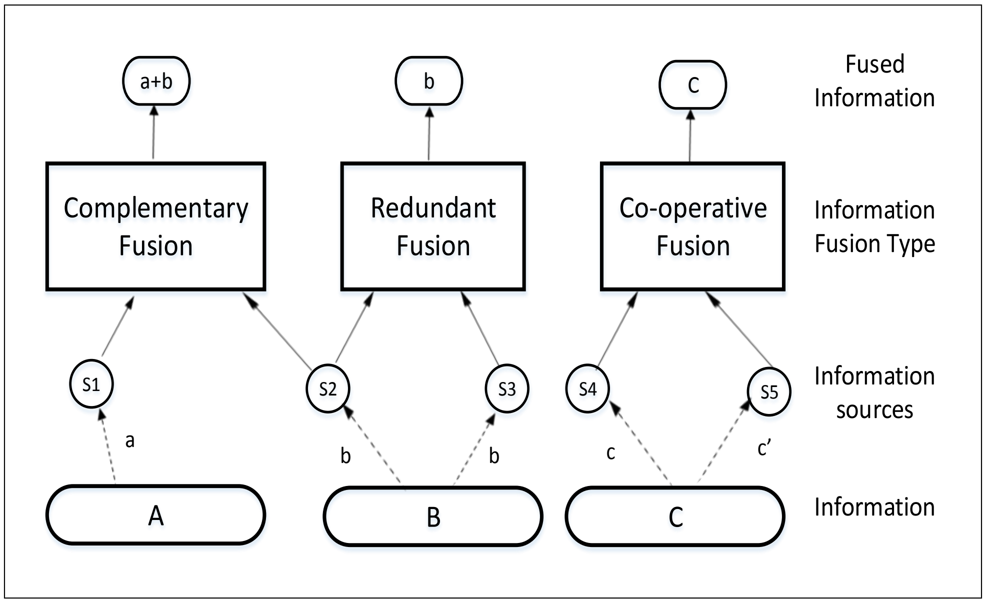


Figure 1. Information fusion classification based on relationship between sources

through the system. Thus, trustworthiness of the source must be verified before information fusion.

- Since there is redundancy associated in the storage phase, when the information is retrieved from various sources, there is likely to be duplication of information. Identification of accuracy of the source has to be verified.
- The keywords in the user query may not be syntactically accurate. However, this should not affect the accuracy of the results obtained.
- Also, there can be multiple interpretations of the keywords associated in the query. It is necessary to determine the context of the query and accordingly perform semantic mining to obtain the accurate results.

An excellent survey by Nakamura et al [19] has described in detail various techniques that can be used for data fusion. These techniques include Kalman filters, Bayesian and Dempster-Shafer inference, aggregation functions, interval combination functions, and classification methods.

Multiple application areas for Information fusion have been discussed earlier. Our work has been driven by making the following observations and identifying the differences in the type of fusion required:

- Most of these techniques have been used for fusion of information collected from various sensors in wireless sensor networks. Sensors hold specific pieces of information only. Hence the fusion activity involves fusion of data like temperature (if they are temperature sensors) or location co-ordinates (if they are proximity sensors) or direction and velocity (in case of accelerometer).

- None of the solutions, to the best of our knowledge, have evaluated the correctness of the query or reinforced and reconstructed it with contextual information to improve the quality of query.

- The sources that have been used for obtaining the data may be malicious or spam. If the trust value has not been considered, the quality of data fusion may be affected.

- When this information has to be obtained from multiple, heterogeneous data sources where parts of information have been stored (and some may contain redundant information), it is necessary to use techniques different from the ones described in the literature.

The problems specified above indicate that we need to handle various aspects in providing a solution, such as:

- Techniques to ensure the correctness of the query itself.

- Techniques to enhance the quality of query based on the context of the query.

- Techniques to search the parts of content from various sources using semantics of the query and mapping them to the existing ontology.

- Techniques to identify the trust value of the source and use cooperation based Information Fusion to present the results.

Our proposed technique Trust based Co-operative Information Fusion that uses NSM algorithm addresses the above issues in the following manner:

- Enhanced N-gram Algorithm:
  Our enhanced N-gram algorithm understands the user query and contextual factors. It clarifies the query request at two levels:
  - Word Level: N-gram algorithm corrects the misspelled keywords.
  - Query Level: N-gram algorithm associates the request with the contextual knowledge to generate a more precise query.
- Semantic and Distributed Machine Learning (SDML) Algorithm:
  SDML performs semantic data mining in a distributed way across heterogeneous data sources. Our wrappers optimize the learning process and our mapper semantically translates the data back to target ontology. During the storage phase, we have assigned a trust value to each of the nodes where the data is stored. The wrappers automatically choose data from nodes with high trust scores.
- Map Reduce-based Fusion (MRF) Algorithm:
  Upon the results from the SDML mining, MRF performs MapReduce-based fusion. It promotes in-depth information fusion and co-operation. The results obtained from a data source can be used as a new clue to learn other data sources and retrieve relevant information. MRF returns the results to the user only if the stop criteria is satisfied, as shown in Figure 19. Otherwise, MRF iteratively exploits collaborative learning and mining until the result is optimal.

Figure 1 shows the various types of information fusion based on the above classifications.

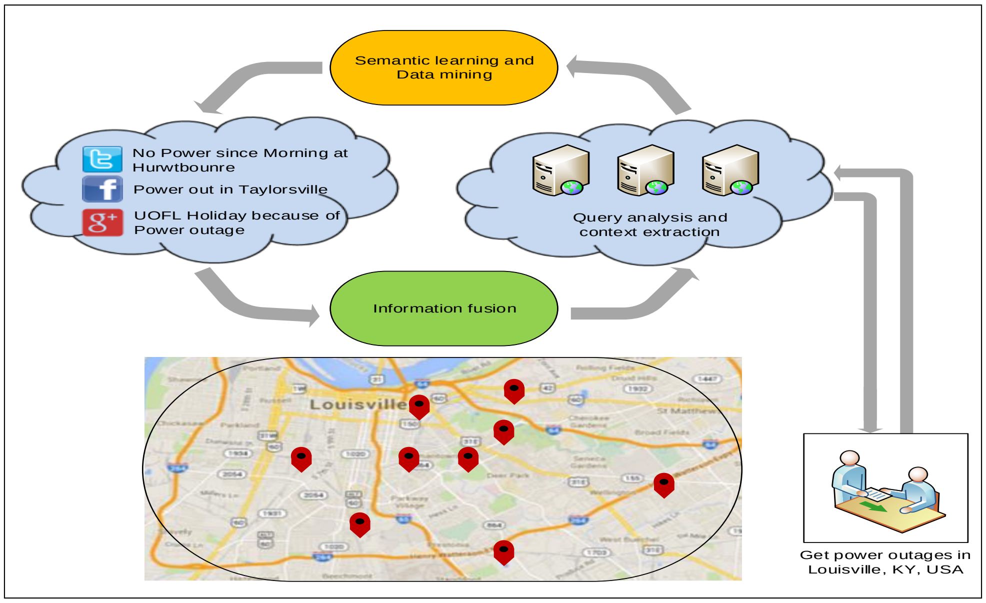

Semantic learning and Data mining
No Power since Morning at Hurwtbounre
Power out in Taylorsville
UOFL Holiday because of Power outage
Query analysis and context extraction
Information fusion
Louisville
Get power outages in Louisville, KY, USA

Figure 2. Information fusion example for power outage example

Figure 2 shows an example of a power outage Information Fusion example. Assume we have a power outage in Louisville and want to know the areas in Louisville that are facing power issues which can be helpful for the electricity department. In this example we first have to understand the user query, perform semantic analysis and data mining on data from various social networking sites like twitter, Facebook, Google Plus like. We process the information and extract the locations of power outages based on tweets and feed. We then use the locations, perform information fusion and display them on a map to show the areas of power outages.

C Complete System

We propose a DDS system on the social network (SDSF) where we build a network of personal storage devices of system users. SDSF promotes the idea of utilizing the storage space of the personal devices along with dedicated storage servers. Figure 3 shows the SDSF system. We encrypt the given file using the user provided encryption key and divide it into smaller pieces. We then add redundancy using erasure coding. Storage nodes are determined from the users' social network. Once

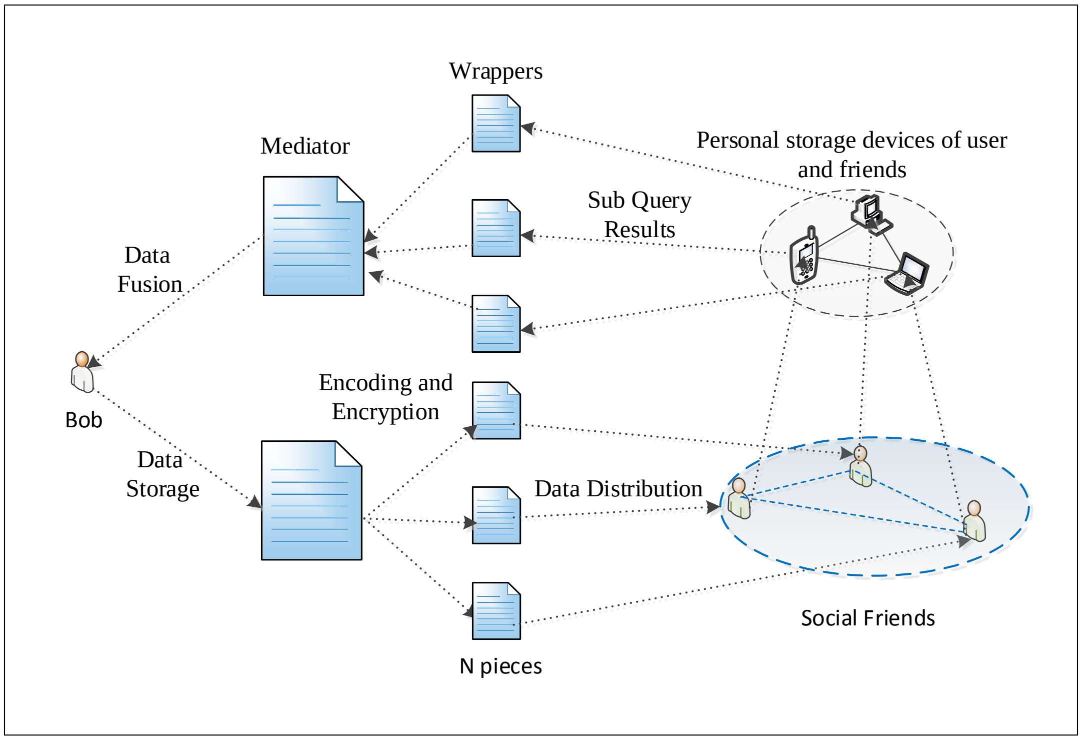


Figure 3. Files can be uploaded by users and can be searched intelligently

the files are uploaded users can search for files with queries. We perform information fusion and return the most relevant results possible. We built a fully functional SDSF system for laptops and desktops and we also discuss the performance of SDSF data storage and information fusion.

CHAPTER II

RELATED WORK

There has been a lot of research in Erasure Coding, Distributed Storage, Social Cloud Computing and Information Fusion.

A Replication

DDS evolved as a means to store data remotely. Although, the number of reads and writes of data determines the strategy to be employed, there are some requirements of persistent storage. The data should be retrieved reliably and should be secure. When the data is on a single server, although the reads / writes are fastest and security is maximum (especially if it is encrypted and saved), the entire data is lost if the server crashes. So, to ensure reliability it is necessary to replicate the data and store it at multiple locations. This implies that redundancy in storage is a must. This replication can be done using additional hardware (RAID) or using software based approaches (creating storage subsystems of the individual nodes and combining them across the network). Examples for such systems are the Parallel Virtual File System (PVFS) [20] and the Lustre Parallel File System [21].

- Full file Replication:
  This method requires that the entire file is replicated and stored at multiple locations based on the nodes that request the files. This strategy works well when the files are small in size. When the files are large in size this strategy is not meaningful since there will be traffic congestion and bandwidth will be consumed with the data transfer. It is also a time consuming process. If the

location of the server where data is replicated requires multiple hops to reach then it might result in a timeout of request and multiple requests may be necessary before the data is obtained. The main disadvantage of using this strategy is that it requires a lot of storage space [22]. If a file of size s is stored as x replicas, the space needed to store the x replicas is $x * s$.

- Block level Replication:

  In this technique, the file is divided in multiple blocks and stored at various nodes. If the data from any one of these nodes is missing, it is difficult to reconstruct the complete file. To avoid this issue, each block also can be replicated. But this will increase the overhead of maintaining the locations of blocks of information and the number of that block in the sequence. Block level replication is used in eDonkey [22].

- XOR Redundancy:

  To introduce redundancy, one may use 1D XOR redundancy or 2D XOR redundancy. The advantage is that if one of the nodes fails, it is possible to reconstruct the original file using the unique property of XOR function that when applied twice in a row it negates itself. The obvious disadvantage is that although retrieval is reliable, security can be compromised. Also when the error correcting capability of the code is broken, there is an infinite error propagation. There are many other variations of XOR operations like Even Odd that allow efficient reconstruction from failures but when the redundancy increases it can become an issue during an update since a large number of redundant data will also need to be updated [23].

- RAID Redundancy:

  RAID 5 is a hardware based file recovery system which needs at least 3 hard disks. It allows failure recovery for a single disk. It internally uses XOR operations to recover the failed disk. It is very fast and efficient. However it cannot handle two disk failures. If two disks fail all the data is lost and cannot be recovered.

RAID 5 with hot spare can be used which allows up to one disk failure. But in this case the hot spare disk becomes active and replaces the failed disk. We then have to replace the failed disk which now acts as the new hot spare disk. It needs at least 4 hard disks.

RAID 6 can tolerate two disk failures at the cost of write speeds. It is slower to write to than Raid 5 but allows up to 2 disk failures. It needs at least 4 disks. We can use raid 6 with a hot spare which will need at least 5 disks.

RAID offers disk failure protection but does not offer any protection for other failures like processor failure or power failure, and the files cannot be accessed unless we move the disks to another machine.

B Erasure Coding

Maintaining high reliability is essential for the data storage systems. Traditionally, data is duplicated at many distributed storage devices for higher reliability. However, this method requires large storage space and consumes a lot of time to maintain a pre-specified level of reliability.

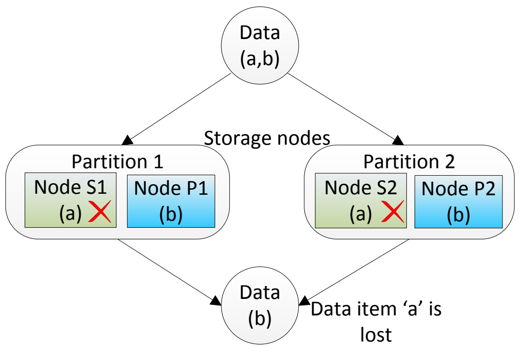


Figure 4. Redundancy using file duplication

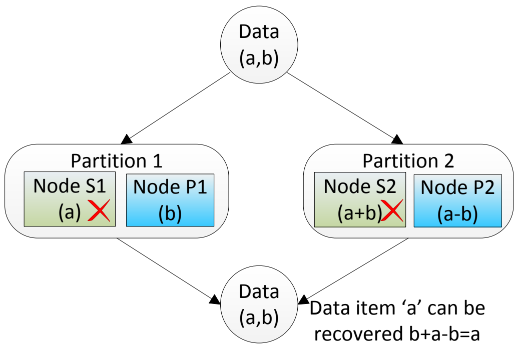


Figure 5. Redundancy using Erasure Coding

Consider we have Data “ab” and four storage nodes to store the data. We split the Data “ab” into two parts nodes “a” and “b”. Figure 4 shows data redundancy using replication. Figure 5 shows data redundancy using erasure coding. Both systems have two data nodes and two redundant nodes. Consider two node failures on each system. In Figure 4 data cannot be recovered as “a” cannot be recovered. However, In Figure 5 data can be recovered from the two remaining nodes. Notably, erasure codes-based approach [24] can provide the same level of reliability with much less required storage space. Particularly, Weatherspoon and Kubiotowicz in their work Erasure coding vs replication [25] provided a quantitative performance comparison in terms of data reliability and transmission bandwidth of the erasure codes-based approaches over the duplication-based approaches. The results showed that a significant performance gain is obtained by using the erasure codes-based techniques. In a different setup, Rodrigues *et al.* [26] showed that in P2P-based storage systems, in some scenarios, the benefits of coding gain over the duplication-based approach is limited. Particularly, the duplication-based approach outperforms the coding-based approach when the node availability is high. The result suggested that performing coding is not necessarily obtaining better system performance. In addition, using multiple measuring metrics,Wang *et al.* [27] showed that the erasure coding based approach obtained higher performance than the duplication-based approach in a cloud system. Interestingly, Leong *et al.* [28] showed that coding is unnecessary when the total size used for storing the file is small.

C Erasure Coding Challenges with Big Data

We are using MDS Reed Solomon erasure codes for Data Redundancy. MDS codes have the (N, K) property where any k of n files are sufficient to recover the entire file. There are two main challenge when using erasure coding for Big Data:

1. The computation time for erasure coding increases with the increase in file size. For a storage system the user cannot be made to wait for too much time. For a

large file, the time taken for normal erasure coding is a lot and is not ideal for the user.

2. The buffer size used in the erasure coding algorithm is bound by an upper limit of the RAM size. The system can only generate some of the available RAM for the erasure coding buffer. So the buffer size cannot be very big. If the buffer size is very small the file has to be opened and closed so many times that it will even further delay the erasure coding as the time taken for I/O operations is very expensive.

Reed Solomon Runtime

When using MDS Reed Solomon erasure codes we have to perform matrix multiplications during encoding which has a runtime $O(n^2)$ and matrix inverse operations while decoding which has a high runtime $O(n^3)$. As the matrix size increases, time taken and computation resources required for matrix inverse operations increases exponentially.

Here we can make use of Map Reduce (explained later) framework to tackle while dealing with big data which breaks a large problem into a subset of smaller problems, solves the smaller problems individually on separate machines and combines the result of the smaller problems.

D Hadoop

Jeffrey et.al. proposed MapReduce: A programming model and its associated implementation for processing and generating large data sets [29]. Map Reduce model uses a map and reduce function. MapReduce runs on a large cluster of highly scalable servers. The servers act as mappers and reducers. Figure 6 shows an overview of the map reduce model. An input file is split into smaller files and is given to the mappers where each file is processed independently and then the results are sorted and given to the reducer. Here the results are combined and merged. Sathiamoorthy et.al.

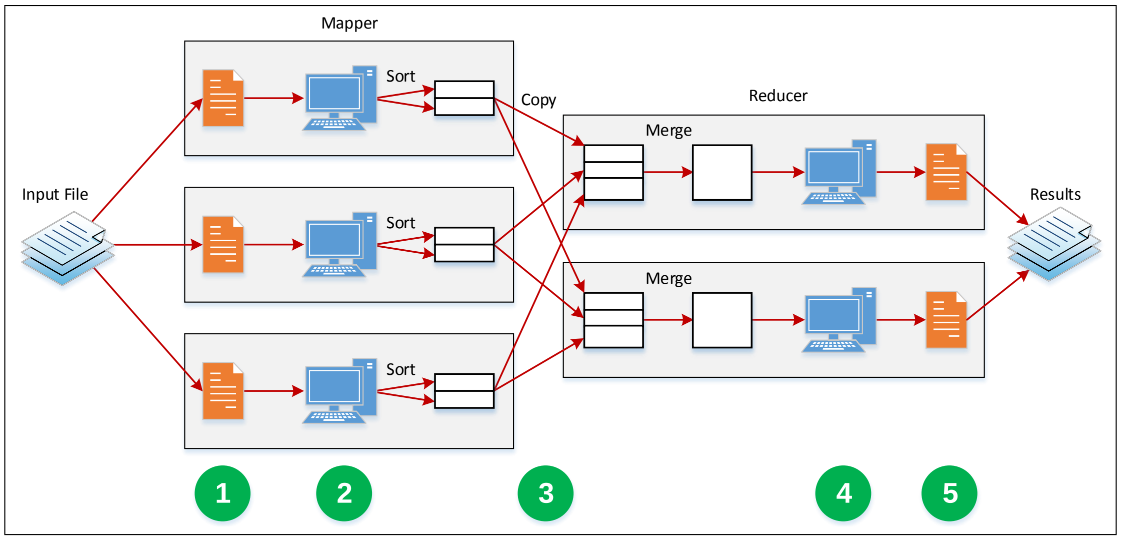


Figure 6. Map Reduce overview

proposed novel erasure codes [30] for big data. They implement their erasure codes in Hadoop HDFS and compares them to Hadoop HDFS with Reed Solomon Codes. They have a 14% penalty in storage space but are very efficient while repairing failed nodes. Xorbas [31] is their Hadoop version that implements a new set of regenerating codes called Locally Repairable Codes (LRC). It is built on top of Facebook's Hadoop system running HDFS-RAID. It supports Reed Solomon and XOR codes in addition to LRCs.

E P2P Systems

Among the large number of known solutions for DDS systems, P2P networks have been used extensively. P2P systems are designed for sharing data among the users. Systems such as Ocenastore [3], Freenet [4], Dht [5], TotalRecall [6] are examples of P2P systems. The main principle of these systems is to allows the users (i.e., peers) to connect directly with each other when sharing the data. Using this approach, data can be shared among users without a central server which could be a single point of failure and can have scalability issues. Bit Torrent File Sharing [32] is another file sharing system where a central server (i.e., tracker) acts as the coordinator for connecting the peers. However, the central server knows nothing about the contents of the files. Thus, the designed system is scalable and is able to support a large number of users. The main issues of the P2P systems for data storage is that these systems provide no guarantee on file retrieval as availability of files depend on the interest of those files by the users. If users stop downloading some files for a certain amount of time, those files will not be available anymore.

Users unwilling to store their data with cloud providers can instead use decentralized P2P networks for their data sharing. Unfortunately, even P2P networks can compromise user privacy. Popular P2P systems not only leak the source of data but also know about which users subsequently access those files. The agents monitoring these networks range from researchers (trying to understand the network properties better) to content providers (trawling for copyright infringement). Either way, users are often unaware of the wealth of information being collected about them when they share or access content over the Internet, irrespective of the fact whether they use a cloud provider or a P2P network. The distributed and global reach of the Internet makes it useful not only for sharing users' personal files but also for business organization needs. The information sharing has become universal but is really vulnerable in terms of security and confidentiality. Governments have shown their interest to selectively restrict the content, source, and destination of communication to benefit their social and economic agendas. The most popular example of one such implementation is the network for Blackberry phones.

In P2P networks, each peer has a specific identifier that is created from hashing the nodes ip address and port number. The placement of nodes and content is predetermined so that while performing a lookup, the a priori knowledge of the location of content is beneficial while performing lookups. Also, the storage has to be persistent since the latency for lookups should be minimal and the content required should be found quickly, along with the requirements of security and privacy for owner as well as the receiver of data. The system Oceanstore [3] also supports nomadic data (disassociation from the location of origin of data), as the authors refer to it. Oceanstore ensures that all the information entering the system is encrypted and only the storage servers have the responsibility of data integrity. Oceanstore stores each data item as an object, with a global identifier, in multiple servers (floating replicas). These objects are updatable and hence can be in active or archived (identifiable through versioning mechanism) which are again stored in multiple servers. A probabilistic algorithm (hill climbing) identifies the location of the nearest active object using the predicate and the desired GUID. The routing is a two phase process using a fault tolerant data structure stored at network layer where messages are routed from node to node till destination is reached. If no active object is found close by, a deterministic algorithm will find the objects far away from the requesting server, which is a slower process. The naming technique provides security from replay attacks to a certain extent. Although this model ensures that the cost of update reduces as the size of the update increases, if the size is small, which is more likely a case, the cost is very high. Also, the latency of update is not known. There is no specific way in which the servers are chosen and so the required redundancy may be quite large. The system uses block level erasure coding for durability and uses simple replication to improve tolerance from transient failures. Using erasure coding and replication, the information is stored in multiple locations, and one assumes that the servers are untrusted, it necessitates the need for unwanted and much more replication than necessary. Erasure coding ensures constant delay but the computation costs are high. Since the systems used in Oceanstore are desktop machines, whose computational capacities are much higher, it may be unsuitable to use the parameters defined for desktop machines in mobile systems as both computational capacities and storage space are challenges in using mobile devices for storage.

In Freenet [4], Clark et. al. have focused on the privacy for the originator of the data and also the node that stores the data and frees the storage node from ownership of the data residing at the node. The protocol used is packet oriented and uses self-contained messages. Each node is treated equally and is aware of only its immediate predecessor and successor. This enhances the privacy since no node is aware if the predecessor node is the originator of request and if the successor node is the consumer of the request. Also a copy of the data is replicated at each node (stored using LRU technique). This improves the lookup time in subsequent queries at nearby locations. While writing the data, it creates a hash key and checks with itself if it has a similar key and also has the required storage space, if not, then it looks up the neighboring nodes and sends data along with the key. If the node finds similar keys, it stores the data, else it forwards it to its nearest node. This also allows creating clusters of data that will reduce the look up time. However, this protocol may be unsuitable in mobile environments since the location of the neighbors will be changing continuously and some nodes may not remain neighbors. Also, the entire file has to be stored. So if an adversary is able to get a file from any one of the participating nodes, the entire data is exposed. This can be detrimental in case of highly confidential information as in case of sensitive data.

In DHASH++ [5] Dabek et.al. have compared the various techniques proposed for designing a wide area distributed hash table (DHT) that provides high throughput and low latency. They have studied various schemes for the design choices of read and write algorithms in solutions using iterative routing (The nodes in the system communicate with only the node from where the lookup has originated, thereby increasing packet overhead), recursive routing (When a query is made, each node forwards the query to the next node, reducing the packet overhead), proximity routing, neighbor selection (chooses the nearest neighbor in the routing table thereby reducing the lookup latency), erasure coding (improves the privacy for the data owner by encoding and introducing redundancy), replication (fragments of data are replicated in multiple servers), and server selection (technique of fetching data from the nearest set of candidate nodes). The experiments demonstrate that recursive lookups increase the lookup time by 1.6 times as compared to iterative lookups. Also the use of erasure coding for data fragmentation reduces the bandwidth consumption for writes, and latency for reads is low, but in case of replication it is exactly the opposite. Based on their observations they have designed an algorithm that tuned the parameters of read and write in a distributed hash tables (DHT). They integrated the key lookup and the data fetch with which they found that they were able to reduce the impact of the lower bound that is imposed by the final lookup steps. Their algorithm also have suggested the use of an alternative integrated transport layer protocol that allows alternate timeouts allowing for freedom to choose from many nodes that are contactable. Although the various modifications have improved the performance, the optimization does not look at how it will affect the high churn network such as in mobile networks. Also, though they have considered the performance parameters of the erasure coding in the design of their algorithm as static in nature, the same parameters may not be relevant in mobile networks where the computational power and available space for storage will be limited.

Bhagwan et.al. in their proposed system Total Recall [6] have focused on predicting the availability of the peers in the network (which has high churn rate in peer to peer networks) using in depth analysis of the a priori knowledge of the rates of peers joining and exiting the network. The system determines the level of redundancy required for fault tolerance based on the file size, storage availability, read/write access rates and uses multiple repair strategies (eager repair or lazy repair) based on the redundancy method (replication or erasure coding) used. The system is dynamic in nature due to these functionalities incorporated; however, it is assumed that all the participating nodes are trusted and the level of trust with each node is same. In reality the originator of request may not trust all the peers in the network. So some mechanism that incorporates the trust component also needs to be taken into consideration.

F Cloud Storage

Internet users share text, voice, images, and videos. Sometimes data is shared with the entire world, sometimes with a few selected friends, and sometimes data is shared in such a way that the original source or destination of the data would preferably stay hidden both from each other and from observers in the network. This shift in usage has made cloud services popular for communication and data sharing. Users upload their data to a centralized provider who then allows the user to share and access the data from anywhere. Centralized cloud storage providers handle everything from video and photo sharing to data collection from simple devices in the home that report water and electricity use. However, using the cloud has its downsides. For example, many popular cloud storage providers collect, store, and share vast amounts of data about their users, raising privacy concerns. Even if we trust the cloud provider with our data, centralization makes censorship easier which has become a serious practical concern in many places around the globe. Cloud storage systems are built on top of a pool of vast number of storage servers. Systems such as Google Drive [8], Dropbox [7], and Amazon Storage [33] typically consist of a few master servers and several storage-servers organized in such a way to efficiently store a large amount of data. These storage systems are able to provide highly accessible and user friendly storage services to the users. Dropbox uses modern encryption methods to both transfer and store the data like Secure Sockets Layer (SSL) and AES-256 bit encryption. Dropbox website and client software are constantly being improved to enhance security and protect against attacks. Two-step verification is available for an extra layer of security at login. You can choose to receive security codes by text message or via any Time-Based One-Time Password (TOTP) applications. Public files are only viewable by people who have a link to the file(s). Once a file is added to your Dropbox, the file is then synced to Dropbox's secure online servers. All the files stored online by Dropbox are encrypted and kept securely on Amazon's

Simple Storage Service (S3) in multiple data centers located across the United States. However, concerns on data security and privacy with cloud storage are main issues due to the unreliability of the service [34, 35]. Generally users upload their files to Dropbox and Google without encrypting their files first. So the administrators of Google and Dropbox can view the user's data without the user's permission. Google collects users' privacy details to target custom ads to the user. So users using these services are compromising on both security and privacy. The below are the Privacy Policies from their websites:

- Google Privacy Policy:
  When you upload or otherwise submit content to our Services, you give Google (and those we work with) a worldwide license to use, host, store, reproduce, modify, create derivative works (such as those resulting from translations, adaptations or other changes we make so that your content works better with our Services), communicate, publish, publicly perform, publicly display and distribute such content. The rights you grant in this license are for the limited purpose of operating, promoting, and improving our Services, and to develop new ones. This license continues even if you stop using our Services (for example, for a business listing you have added to Google Maps). Some Services may offer you ways to access and remove content that has been provided to that Service. Also, in some of our Services, there are terms or settings that narrow the scope of our use of the content submitted in those Services. Make sure you have the necessary rights to grant us this license for any content that you submit to our Services.

- Dropbox Privacy Policy:
  When you use the Service, we automatically record information from your Device, its software, and your activity using the Services. This may include the Deviceś Internet Protocol "IP") address, browser type, the web page visited before you came to our website, information you search for on our website, locale

preferences, identification numbers associated with your Devices, your mobile carrier, date and time stamps associated with transactions, system configuration information, meta-data concerning your Files, and other interactions with the Service.

TABLE 1. Cloud Storage Cost Comparison

Size/Cost	Google Drive	DropBox	Amazon	SDSF
2 GB	Free	Free	Free	Free
5 GB	Free	N/A	Free	Free
15 GB	Free	N/A	N/A	Free
1 TB	9.99$	9.99$	41.66$	Free

Another concern with Dropbox and Google Drive is the complexity of the system. As the system size expands, the complexity of the system also increases. Also personnel required to manage hardware and software increases. Another major concern is the cost. Google Drive offers 15 GB free of cost and Dropbox offers 2 GB free of cost to the users while Amazon Drive offers 5 GB of free storage. Users have to pay to get additional storage space in cloud systems even if they have a lot of free storage space in their laptops, desktops, tablets and mobile. The associated costs are compared in Table 1. In SDSF we use the free space of the user's storage space to build a free social cloud storage service.

G Social Networks

Social networks like Facebook [36] and Twitter [37] are growing rapidly with large number of users. Further, many techniques have been proposed to quantify the social relationships based on the social interactions in the social networks. These works offer a solid foundation to characterize the social ties, which are exploited in our proposed system. The work that is most closest to ours is Safebook [38]. In this work, the authors presented a system design for data storage and management to preserve the privacy, integrity, and availability of the stored data. However, this work does not deal with the reliability and efficiency (e.g., erasure coding in our SDSF), and also does not provide any analytical analysis as well as an efficient method to quantify the social ties. Chard *et al.* proposed approaches [39, 40] for sharing storage and computation resources among users utilizing the social relationships. Particularly, the authors used the social relationships to quantify the trust between the users to create a dynamic virtual connected network. Again, the work did not provide any analysis implementation or results. Different from these systems, our proposed system SDSF provides a comprehensive system design with extensive analysis and simulations. Importantly, we implemented a real network test bed for system validation.

Table 2 compares the above systems. Based on the above discussion, we present the summary to provide a comparison of various storage systems. The parameters that we considered to make a significant contribution to the "storage" are quantum of data that can be stored, the ease of data management, the complexity of creation of data storage system, and the security and privacy. Table 2 gives a bird's eye-view of comparison between various storage systems.

H Information Fusion

Information fusion is extracting and learning useful information based upon the user's query in a given context. It is mostly performed by machine learning and other reasoning from a given set of data. Information fusion becomes a challenging problem when the data are stored at distributed storage devices and the trustworthiness of the source is not taken into consideration. Information fusion in the cloud has further challenges that should be further addressed:

- Heterogeneous Data Formats:
  Data created and maintained by different organizations often have different formats and some data may not be well structured.
- Partial and Distributed Data:
  A data source may only possess a partial information of some real-world data objects. Therefore, it is a problem to integrate the partial information from different cloud sources to obtain the accurate and complete reasoning.
- Distributed Processing:
  The data in the cloud is huge and is beyond the computational capability of a single machine. The question is how to design a information fusion engine to reason and reference belief nets of data patterns across the cloud nodes.

Recently, Hull *et al.* proposed an approach called WAVE-FE [41] for semantic enrichment and fusion for multi-intelligent data. WAVE-FE adds semantic metadata tags

TABLE 2. Comparison of Storage Systems.

	Storing on PC	P2P Network	Dropbox	SDSF
Reliability	Data is lost when system crashes	Multiple copies of files are stored in different places.	Uses redundancy and version control	Uses Erasure coding for redundancy.
Security	If system is compromised, entire data is unsafe.	Files can be accessed by others.	Files can be accessed by Dropbox administrators	Need to compromise k out of N devices to retrieve data.
Privacy	Privacy information is not exposed.	Privacy information is not exposed.	Collects user's privacy details.	Privacy information is needed for context but does not associate them with their identity.
Storage Size	Fixed.	Storage size increases as P2P users increases.	Storage size can be increased with cost.	Storage size increases as users increase, as they share their resources
Data Management	Third Party applications to share data.	Data deletion and controlling access is difficult	Done by Dropbox UI	Done by Social DDS UI
System Complexity	System maintained by user	System expands and is managed by individual users	System complexity increases with users	System expands and is managed by individual users

to the original data enabling advance correlation and fusion. Hadoop and MapReduce are other approaches that are developed to increase the fusion capabilities on the cloud. However, these approaches are still evolving and do not meet the requirements of information fusion and they have some challenges:

- Fuzzy-enabled Query:
  For some critical applications, users may not be able to accurately query the information (e.g., keyword typos) in a time intensive or unknown environment. On the other hand, users may lack the contextual knowledge to perform an accurate query. The fuzzy-based query on the information fusion means the capability to provide the right information to users even if the query is not accurate.

- In-depth Fusion:
  The storage system needs in-depth information fusion to really understand the needs of the users. Given a query from a user, in-depth information fusion not only provides the capability to merge heterogeneous data, but also conduct collaborative reasoning across these data sources to give the most appropriate information desired by the user.

- Fusion Precision:
  High fusion precision is the key for tactical information fusion services. Inaccurate, biased, or wrong fusion results may lead to wrong actions and even mission failures. The query precision should be based on all accessible data resources.

- Real-time Response:
  Delayed response could result in tactical action delay or loss of actions. Therefore, the information fusion should be efficient to process all data in a time-constrained manner. Massive information exchange among nodes for reasoning may not be applicable for storage services as this creates significant delays.

CHAPTER III

STORAGE

A SDSF Storage Framework

The high level architecture is shown in Figure 7. There are 6 main modules in

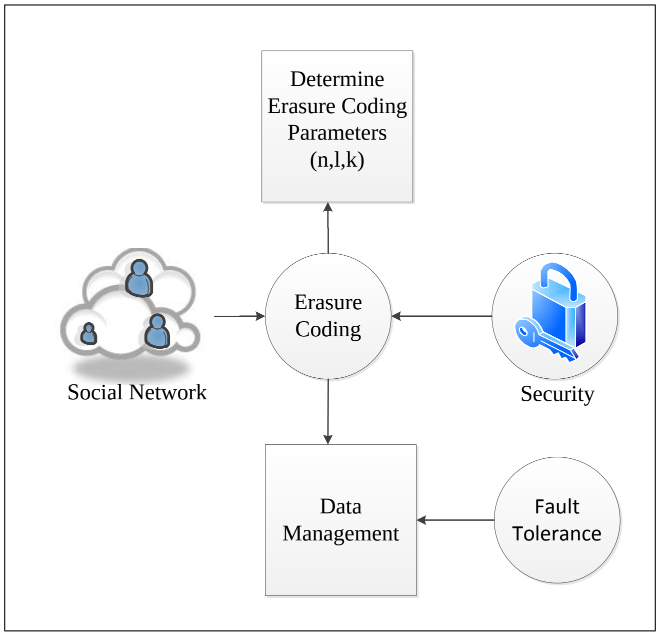


Figure 7. High Level Overview

the system:

1. Erasure coding module
2. Determining erasure coding parameters
3. Social Network module
4. Fault Tolerance module

5. Security module

6. Data Management module

The Erasure coding module helps in maintaining the reliability and security of the SDSF system. Erasure coding is performed on the individual file where the file is split into multiple pieces and is stored on the storage devices. The SDSF system then determines the erasure coding parameters based on the file size, network speed and other parameters like available storage space. The erasure coding parameters play a vital role in providing security and reliability. The potential storage devices are then identified from the social network module. The Data Management module is used to read, share or delete files. The Fault tolerance module is used to recover data when storage nodes fail. When a user logs in with his credentials, he is presented a screen where the users' files are displayed. The users can view, share, upload, download or delete the files. The Security Module takes care of authentication, authorization, encryption of data and other security features.

1 Design Goal for Users

The detailed system view is shown in Figure 8. The Erasure coding module is explained in Section B. Determining the Erasure coding parameters is explained in Section C. Social ties and trust is explained in section E. Exploring social storage provider module is explained in section E.1. Fault handling module is explained in section F. When a user wants to upload a new file, the user is first asked to enter an encryption key. We encrypt data using AES encryption with the user assigned key and transmit the encrypted data to SDSF using SSL safely. This ensures that even the administrators of SDSF cannot view the user's data. Once the encrypted data is uploaded to SDSF, we determine the erasure coding parameters and split the file into multiple pieces, and we use erasure coding for replication. We store some of these pieces on the user devices and remaining pieces on the storage devices of

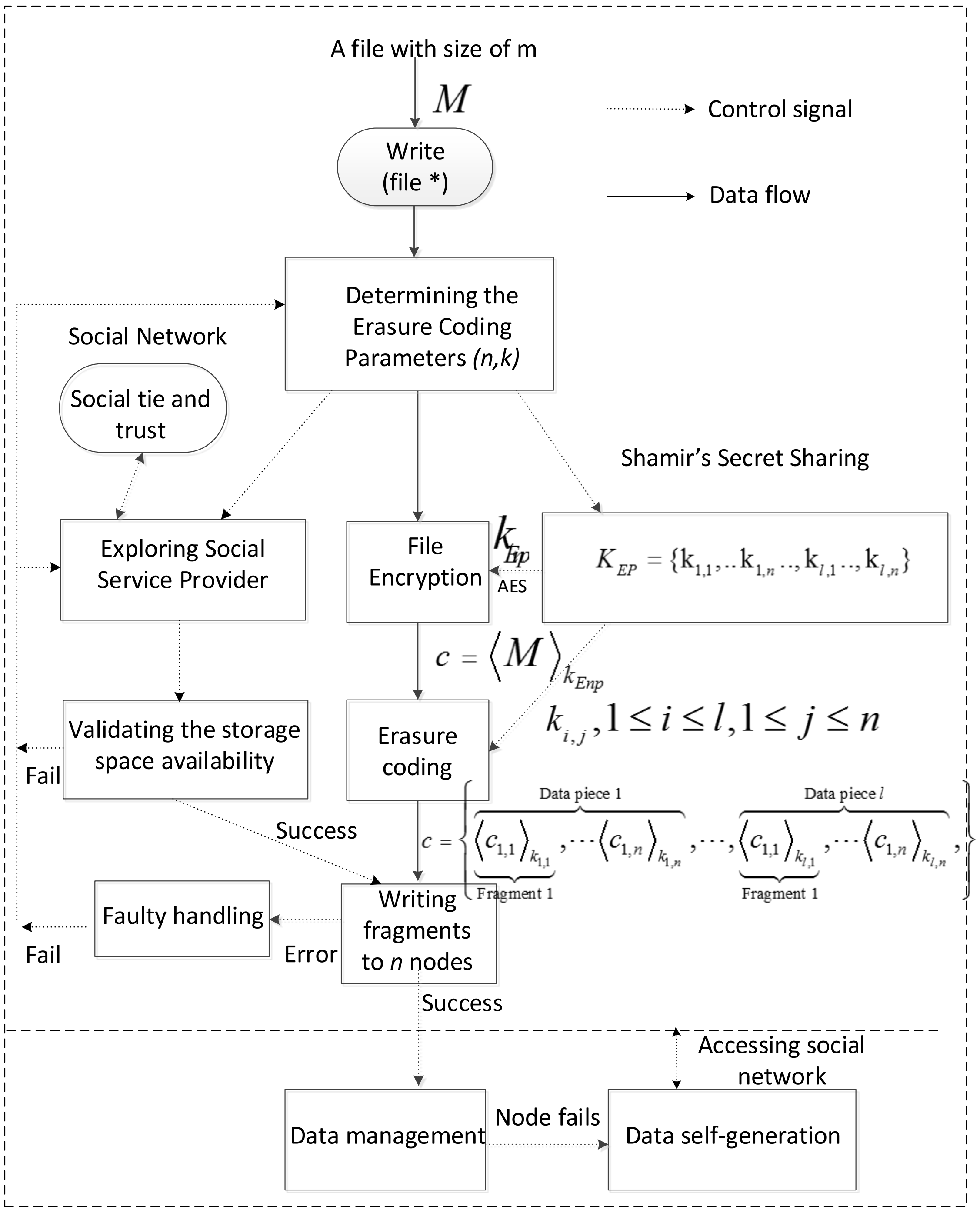

A file with size of m
M
Control signal
Write (file *)
Data flow
Determining the Erasure Coding Parameters *(n,k)*
Social Network
Social tie and trust
Shamir's Secret Sharing
Exploring Social Service Provider
File Encryption
k_{Enp}
AES
$K_{EP} = \{k_{1,1},..k_{1,n}..,k_{l,1}..,k_{l,n}\}$
$c = \langle M \rangle_{k_{Enp}}$
Validating the storage space availability
Fail
Erasure coding
$k_{i,j}, 1 \le i \le l, 1 \le j \le n$
Success
$c = \left\{ \overbrace{\underbrace{\langle c_{1,1} \rangle_{k_{1,1}}}_{\text{Fragment 1}}, \cdots \langle c_{1,n} \rangle_{k_{1,n}}}^{\text{Data piece 1}}, \cdots, \overbrace{\underbrace{\langle c_{1,1} \rangle_{k_{l,1}}}_{\text{Fragment 1}}, \cdots \langle c_{1,n} \rangle_{k_{l,n}}}^{\text{Data piece } l}, \right\}$
Faulty handling
Writing fragments to *n* nodes
Fail
Error
Success
Accessing social network
Data management
Node fails
Data self-generation

Figure 8. Detailed system view

the user's friends and family as determined from the social network module. Social storage providers are determined by the social ties and trust. There are two levels of protection for the users' data. First, the user has to be authenticated and authorized to access the data. Second, the user must know the encryption key, which was used while uploading the file. The advantage of using erasure coding is that any k out of n servers can recover the entire data. The advantage of using distributed storage is that a malicious user must obtain access to at least k out of n devices to retrieve even the encrypted data. We store the pieces of information on the user's friends and personal acquaintances storage devices instead of storing randomly like in P2P networks. We calculate the trust and other parameters like availability, bandwidth, etc. from the users' social network and find the best storage nodes available. This is explained in more detail in section E Evaluating Potential Storage providers. In case some storage nodes fail, data can be recovered from the remaining nodes which is explained in more detail in section F Fault Tolerance.

2 Architecture

Our proposed system consists of three layers as shown in Figure 9:

- The physical layer:
  The physical layer is the Internet layer which offers the social devices' connectivity.

- The social interaction layer:
  The social network layer over the Internet offers the capability for SDSF to distinguish the device's reliability and trustworthiness. This additional layer differentiates our proposed SDSF from the P2P systems where all devices are treated with equal priorities for data storage. SDSF advocates the social network and the devices are organized in a manner based on social interactions. As a result, every social user could be a service provider to offer storage space.

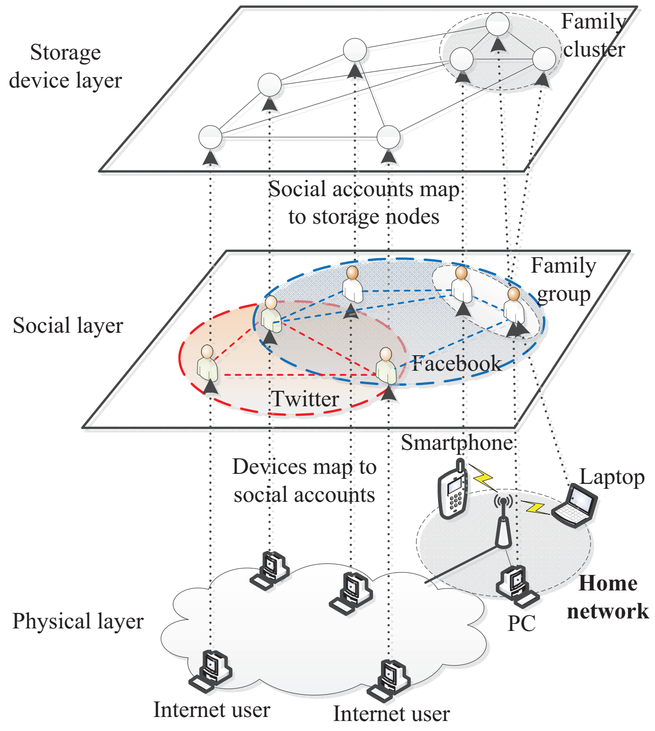


Figure 9. The proposed SDSF architecture.

- The storage device layer:

  The storage device layer consists of various storage spaces of different spaces provided by user mobile terminals and are mapped to the social account. The user's cloud space can also be considered as a part of storage space. In this way, the data could be stored on the cloud as if it is a storage device.

B Erasure Coding

1 Overview

Erasure codes have been around for a few decades and have been used for multiple applications. Erasure coding's key advantage is that it expands the original data by adding a parameterized amount of computed redundancy so that even if a subset of the data is lost or corrupted; it is still possible to retrieve the original data. All data can be recovered if and only if k out of n distributed chunks can be accessed. Consider n to be the number of storage nodes used for storing a file. The data file is encoded into n fragments. Let k be the minimum number of fragments to reconstruct the original data piece. For simplicity, we use (n, k) to represent the erasure code where a file is encoded into n fragments, and k is the minimum number of fragments to recover a data piece. The MSD erasure codes provide a space-optimal data redundancy against node compromises. The file size does not matter for erasure coding as we perform erasure coding multiple times on a file processing small pieces at a time. When the file becomes larger, we still perform erasure coding on small pieces, multiple times, one at a time. So however large a file is we are still able to perform erasure coding. It is desirable for performance to maintain the property that the size of each fragment does not exceed the size of the original file. The erasures codes on each piece generate n encoded fragments by the coding parameter k. The file will totally generate $n.l$ fragments if l is the size of the data we perform erasure coding at a time. These fragments from a piece have to be stored into different nodes. Therefore, n is also the minimal number of storage nodes. Any k fragments out of n are enough to reconstruct the entire original file. The SDSF system is able to recover the simultaneous failures of up to $(n-k)$ storage nodes. In other words, (k) indicates the security factor and $n-k$ indicates the reliability factor which should match to the security and reliability levels of the given file. The size of each fragment is:

Figure 10 shows the implementation of the erasure coding algorithm. The data string c is first equally divided into k segments (e.g., $c = (c_1, \cdots c_k)$). The erasure coding adds $(n-k)$ redundant segments (i.e., $c'_{k+1}, \cdots, c'_n$ in the encoded string as shown in Figure 10). These redundant segments are computed by a polynomial function. In this polynomial function, these k segments are converted into coefficients that are integers (i.e., ASCII strings of the segment to integers). The increase of the size of the data string generates huge coefficients (i.e., huge integers). Because of this, the evaluation of the polynomial function becomes very hard, resulting in very high computational complexity that cannot be accommodated for the mobile devices.

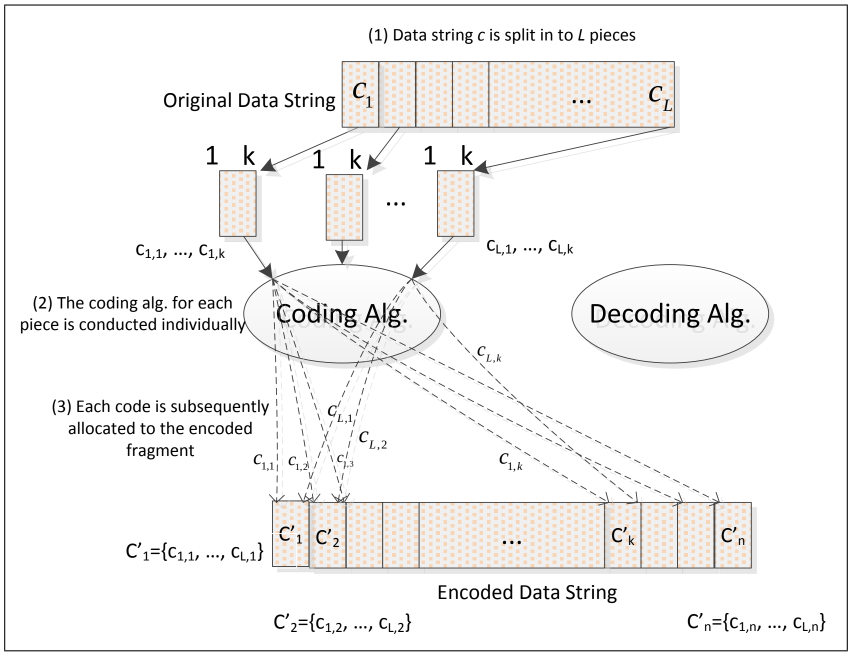

(1) Data string c is split in to L pieces
Original Data String
c_1
...
c_L
1 k
1 k
1 k
...
$c_{1,1}$, ..., $c_{1,k}$
$c_{L,1}$, ..., $c_{L,k}$
(2) The coding alg. for each piece is conducted individually
Coding Alg.
Decoding Alg.
(3) Each code is subsequently allocated to the encoded fragment
$c_{L,k}$
$c_{L,1}$
$c_{L,2}$
$c_{1,1}$
$c_{1,2}$
$c_{1,3}$
$c_{1,k}$
C'$_1$
C'$_2$
...
C'$_k$
C'$_n$
C'$_1$={$c_{1,1}$, ..., $c_{L,1}$}
Encoded Data String
C'$_2$={$c_{1,2}$, ..., $c_{L,2}$}
C'$_n$={$c_{1,n}$, ..., $c_{L,n}$}

Figure 10. Classical Coding Algorithm

Conversely, the erasure decoding algorithm decodes any k of n segments into the original data string. It also has high computational complexity. Therefore, we have to select the erasure coding which will have optimal performance without sacrificing security and reliability.

C Determining erasure coding parameters

The coding parameters n and k affect security, reliability and performance of the system. Security and reliability are inversely proportional to each other. As the coding parameter k increases, reliability decreases and security increases. The parameter k means it needs at least k pieces to reconstruct the original file. So as k increases, the intruder needs to get access to more file chunks, as a result making it harder for the intruder to get access to the original file. But also as k increases, it needs more chunks to reconstruct the original image, decreasing the reliability. As k increases the runtime decreases. Though the runtime decreases with increase in k, we do not want a high k, as it decreases the reliability. As n increases, the runtime increases, decreasing the performance of the system. We have to find a balance of reliability, security and performance.

From our experiment results we have found that the (n, k)=(8, 4) codes worked ideal for us with a balance of reliability and security and better performance. It is also shown in the performance results section.

D Social Network

Digital relationships are a virtualization of real world relationships between individuals. These relationships infer a level of trust between users. For many of us, social networking has become a primary mode of communication among family, friends and coworkers. This model is used for virtualized resource sharing in a trustful environment. Social Cloud thereby allows users share heterogeneous resources within the context of a social network. The participants of a community in a social network are bound by fixed resources and limited capabilities. Among friends, surplus storage availability, if shared, could be used to meet the dynamically fluctuating demand of other participants. A Social Cloud takes advantage of pre-established trust relationships among the users, which enables mutual sharing. It is similar to sharing resources with our friends in our daily lives. In a similar manner, this concept lets us share our storage resources. Social cloud is not a representation of a point to point data exchange; rather it can be picturized as a multipoint sharing within a group. A social layer can reside over an existing architecture and keep every participant collaborated. Storage perhaps is the most simplest and standardized resource for daily users to share and utilize data from across the globe. Online data storing devices are primarily used to store, create backup, share, and replicate data in one or the other form. One obvious use for social storage is sharing photos and videos on social networks which carry the responsibility for hosting them as well. Instead, it shall prove to be a brilliant idea if this responsibility could be shifted from the storage provider to all the users of the storage system for a better scalability and decreased infrastructural requirements. In order to share resources we need to keep a track of the resources of the friends and acquaintances. We however cannot maintain the resources of each persons along with the relationships on a single machine. In the next section we will see how to maintain the list of resources even for a large scale social network like Facebook.

1 Social Network Resource Management Implementation

When implementing a social network one will deal with millions of data. This large number of users can cause difficulties which will be discussed in the following section. We first start our design with a simple case and we then apply it for a larger network. We first build a graph by assuming every storage machine is a node. Each user can have multiple storage machines like phones, tablets, laptops and computers. An edge exists between two nodes if the two people are friends with each other. We can design a class for the node as

```
public class Node
 {
      private long NodeID;
      private Node[] friends;
      Node(private long ID)
      {
          NodeID=ID;
      }
      public Node[] GetFriends();
      public bool AddFriend(Node node);
      public bool RemoveFriend(Node node);
}
```

If we want to find the connection between two people we can use the breadth first search algorithm.

```
Input: A graph G and a root v of G

procedure BFS(G,v) is
    create a queue Q
    create a set V
    enqueue v onto Q
    add v to V
    while Q is not empty loop
        t      Q.dequeue()
        if t is what we are looking for then
            return t
        end if
        for all edges e in G.adjacentEdges(t) loop
            u      G.adjacentVertex(t,e)
            if u is not in V then
                add u to V
                enqueue u onto Q
            end if
        end loop
    end loop
    return none
end BFS
```

But the problem comes into picture when we are dealing with millions of users. We will not be able to store data of millions of users on a single machine. The data has to be distributed onto multiple machines and our simple implementation above will not work.

2 Large Scale Design

When we deal with Big Data, like the data of social networks e.g., Facebook, Google+, Twitter etc., we cannot possibly keep all of our data on a single machine. It means that a simple data structure will not suffice. It is explained in more detailed in the book "Cracking the Coding Interview" [42]. Data for different persons may reside on different machines or even different data centers. Hence we need a better method of searching for individual data and relationships data across multiple machines. This can be described by an algorithm, described in C# code below.

The following code demonstrates our algorithm:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SocialNetwork
{
 public class WebServer
 {
  public List<Computer> Computers = new List<Computer>();
 }

 public class Computer
 {
  public List<Person> personDirectory = new List<Person>();
  public int computerID { get; set; }
 }
```

```
public class Person
{
 private List<int> relationships = new List<int >();
 public int personID { get; set; }
 private int machineID { get; set; }
 public string Name { get; set; }
 private WebServer Web_Servers = new WebServer();
 public Person(int iD, int machineID, string name)
 {
  Name = name;
  personID = iD;
  this.machineID = machineID;
 }
 public int[] getRelations()
 {
  //Method 1
  //Create an array with same size as the
  //relationships list of a person
  int[] relationsArray = new int[relationships.Count()];
  //Copy ids of related persons to array from the list
  relationships.CopyTo(relationsArray);
  return relationsArray;

  //Method 2
  // return relationships.ToArray();
 }
```

```
//Add relation to person data based on provided ID
public void addRelation(int id)
{
 relationships.Add(id);
}

// Search for a person given their person ID
// and computer ID
public Person searchRelation(int _computerID, int _ID)
{
 //Retreive computer by ID from Web Servers
 // Computers collection
 var Target = from computer in Web_Servers.Computers
  where computer.computerID == _computerID
   select computer;

 //Method 2 - Optimization, use already provided function
 // var targetComputer= lookupComputer(_computerID);

 //Cast to Computer
 var targetComputer = (Computer)Target;
 //Find person in Person directory if the person exists
 var person = targetComputer.personDirectory.
  Find(p => p.personID == _ID);
 if (person != null)
  return person;
 else
  return null;
```

```
    }

    // Search for computer given computer ID
    public Computer searchComputer(int _computerID)
    {
     //Find computer given the computer ID
     var computerResult = Web_Servers.Computers.
      Find(x => x.computerID == _computerID);
     if (computerResult != null)
     {
      return computerResult;
     }
     else
     {
      return null;
     }
    }
  }
}
```

Following are some optimizations that can be applied to our algorithm to improve multi node searches:

- Reduce Hops:

  Hops from one machine to another are computationally expensive when searching through the entire graph. As such we need a mechanism in which multiple related persons are located in the same node (computer) or if one relation of person is found on a single computer the next should also be searched in the same node. This would reduce machine hops which in turn would speed up the search through the graph.

- Use of a hash table for visited nodes:

  In a usual traversal, visited nodes are marked. This cannot be used in our case since simultaneous searches might be done, hence the visited flags may cause unexpected results. For optimum performance and to avoid collision with other searches, we can use a hashtable of visited nodes. Each search will have its own hashtable, hence there will be no collision and traversal can occur in a efficient manner.

- Redundancy:

  Redundancy is key for Social Networks. If a system fails there should be another system ready to take its place. Redundancy will also be used for sharing workload. There can be multiple computers that work in parallel which share their jobs, and if one stops the others can share the workload of the failed machine. Same data is spread across multiple machines to cater for server failures and for parallel processing.

- Collection of Related People in a single machine

  It is much more probable statistically for people to have relationships within the same country. Using this knowledge, we can group people of same countries on machines in the same geographical location. This procedure can be done on addition to the datastore. In addition, we can check locality of the person and add to the respective machine. This should help us achieve faster response times during searching and traversal for person data.

- Caching Data per Region

  We can also reduce search times by caching person data based on searches. Every search should first search inside the cache instead of hopping multiple machines. The cache could be considered a hashmap for sake of simplicity and every search would first go through this. This could drastically reduce search time as the cache can be located on the web server instead of accessing the database. Data can be processed in parallel by several concurrent systems and this is the key to the speed of large social networks.

E Evaluating the potential social storage provider

SDSF stores the data based upon the underlying trust in the nodes from the social networks unlike the untrusted nodes in P2P. Social networks are built based on real world relationships and can be used to form virtual organizations as shown in Figure 11. We can use the different social networks like Facebook, Twitter etc. to find the relationships which can then be used to form a virtual organization.

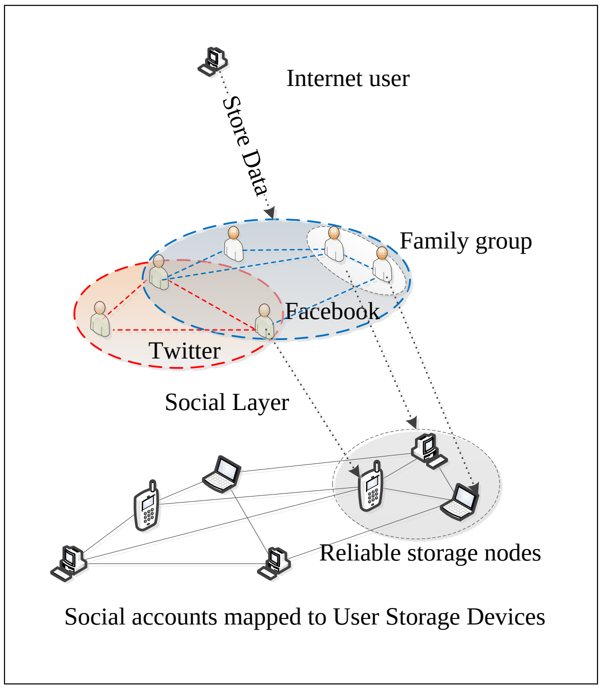


Figure 11. SDSF using Social Networks

Social network is a weighted directed finite graph

$$G = (V, E) \tag{1}$$

consisting of a set V of nodes (friends, family, colleagues and other acquaintances) and a set E of edges (relationships). Each node is characterized by the tuple

$$< S, A, B, Dg, D > \tag{2}$$

where S refers to storage space available on the node for storage allocation, A refers to availability percentage which is the ratio of on-time to total-time, B refers to bandwidth of node, D_g refers to Degree of node, and D refers to the delay. To locate the potential storage nodes, the source node explores the social storage provider. In particular, the source node will search for social nodes that have enough storage space while satisfying certain constraints such as trustworthiness and availability. In practice, however, the size of the social network is very large (i.e., millions of nodes). It is impossible to search all the options to find an optimal solution. Randomly selecting n nodes out of the available nodes could result in an undesirable outcome such as stored data may not be recoverable. In the following subsections, we will show how a source node optimally selects storage nodes. While selecting nodes for storage we have to take into consideration the following:

- Security of data
- Time to perform operations on files
- Availability of files

The evaluation of these attributes along with trust is used to identify nodes that are trust worthy for storing data. We have to identify a potential subset $v \subseteq V$ in order to store the user's data. At the same time, the system must be flexible to allow users to specify additional parameters for determining storage nodes.

Trust relationship between nodes is one of the criteria in the selection of the storage providers. Trust t in our context is the confidence a user i has in another user j that his data will not be misused in any way by the user j. Let $t_{i \to j}$ be the trust or confidence, node i has in node j. Assume $t_{i \to j} \in [0, 1]$.

Obtaining the global optimal solution is impossible due to the large size of the social network. Therefore, our approach seeks for a local optimal solution via an iterative-deepening search on the social network provider. The purpose of this step is to identify a set of nodes V_s which are trustworthy and have sufficient storage space. The search algorithm is illustrated in Figure 12.

1 Finding the Potential Storage Nodes

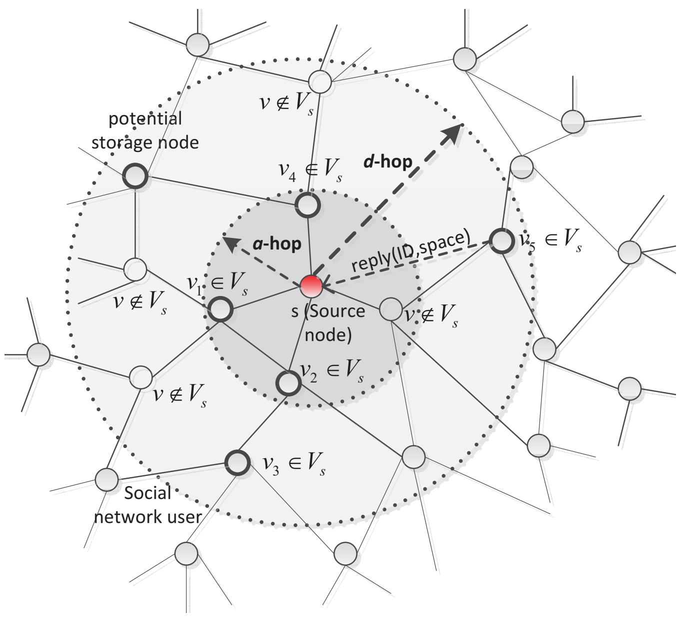


Figure 12. Exploring the social network provider to identify the potential storage nodes.

The source node s (i.e., the node wants to store data) performs many breadth-first searches (BFS) with the depth enlarged after each iteration, until the query is satisfied or the maximum depth d has been reached. For example, s wants to have a two-iteration deepening request where the first iteration searches for potential storage nodes within the depth of a hops and the second iteration searches for potential storage nodes within the depth of d hops ($d > a$). Nodes (i.e., social network users) having the requested storage space within a hops away from the source will reply to the source. Upon receiving replies from the storage node, the source s will evaluate if the request is satisfied (the number of potential storage nodes is greater than n). If not, s proceeds to another BFS of depth d by sending a signal to the a-hop nodes forwarding the request message for $(d-a)$ more hops. Using this approach, the source will progressively improve its search result by increasing the depth of the search after each iteration. To mitigate the effect of the low-rate transmission links, the source can change its time window T to a larger value to accommodate delayed replies. In the next subsections, we will describe how the source node computes the social ties with respect to the potential storage nodes.

2 Evaluating Trust Score of Neighboring Nodes

Two nodes are considered neighbors if they are either directly connected in the social networks or have interacted with each other in the past.

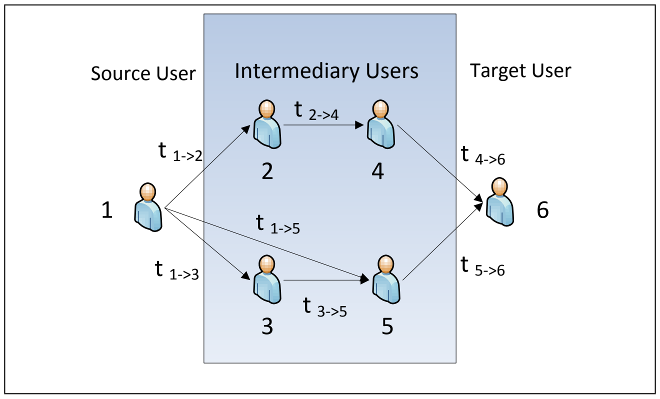
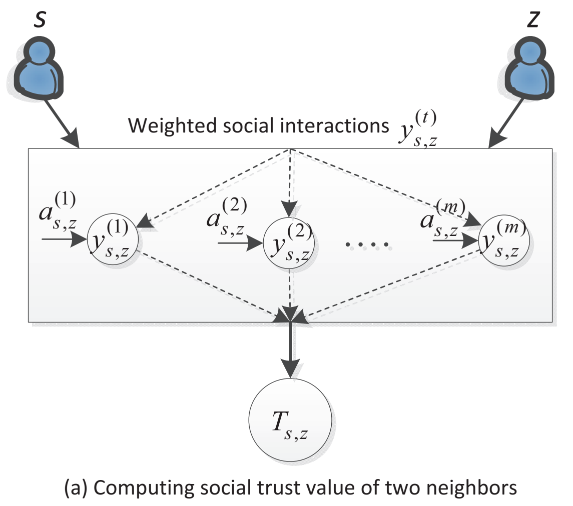


Figure 13. Social Network of User 1. Users 2, 3 and 5 are direct friends of User 1. Users 4 and 6 are indirect friends of user 1 through common or intermediary friends

A sample social network is shown in Figure 13. Users 1 and 2 are considered neighbors and users 1 and 6 are considered non neighbors. A simple way to select storage nodes is to use "service trust" between neighbors. The "service trust" value indicates the satisfaction of the data storing service of a source node on a data storage node. For example, node A stores its data into node B and B always correctly responds to A when requested. Thus, A trusts in B and sets its service trust value in B to a high value. However, selecting storage nodes solely based on the "service trust" in distributed storage system may not reliable as a malicious node can dissemble working properly before turning into an adversary node. To cope with this, in SED, a source node selects its storage nodes based on a trust score which is computed from the service trust and social trust values. The social trust value of node s on node z indicates the trust level of node s on z from the social viewpoint. For example, node s has social relationships with A and B, where A is a family member and B is a classmate; thus, the social trust of s on A is generally higher than that of s on B.

- Social Trust Value:
  The social trust value is derived from the online social interactions (e.g., profile viewing activities, picture tagging, wall status postings, etc.) between the users. The stronger the relationship (i.e., higher social trust value), the higher the likelihood that a certain interaction takes place between the users [21]. Assume that s and z are two neighbors and let $y_{s,z}^{(t)}, t = 1, ..., m$ be the m different types of interactions between them. We define $T_{s,z}$ as the social trust value of s on z. Here, for simplicity, we assume that the trust value of s on z is the same as that of z on s. However, the framework can also be extended for the scenario of unequal trust values between two neighbors, albeit with additional computation. To compute $T_{s,z}$, time is divided into several equal periods, i.e., time windows $[t_{i-1}, t_i]$. Let $n_{s,z}^{(t)}(i)$ be the number of interactions of type t in the social network between s and z during the ith time window, and $\Delta t_{s,z}^{(t)}(i)$ be the time span of type t interactions. Furthermore, let $a_{s,z}^{(t)}(i)$ be the auxiliary factor capturing the impact of type t of interactions between s and z in the ith time window. The graphical model of social trust computation is shown in Figure 14(a). Mathematically, the social trust value for the ith time window can be computed as:
  $$T_{s,z}(i) = \sum_{t=1}^{m} a_{s,z}^{(t)} \frac{n_{s,z}^{(t)}(i)}{\Delta t_{s,z}^{(t)}(i)}. \tag{3}$$
- Service Trust Value: On the other hand, the service trust value is derived from



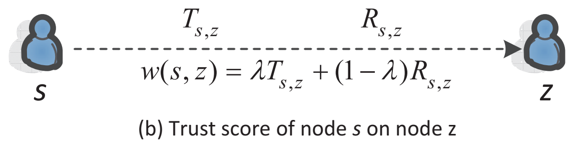

$T_{s,z}$
$R_{s,z}$
$w(s,z) = \lambda T_{s,z} + (1-\lambda) R_{s,z}$
s
z
(b) Trust score of node *s* on node z

Figure 14. Graphical representation of trust score of two neighbors.

the past experience of the data storing service of node s on z. Let $g_{s,z}^{(t)}(i)$ and $\tau_{s,z}(i)$ be the number of satisfied storing service transactions and total transactions of s to z in the ith time window. The service trust value is expressed as:

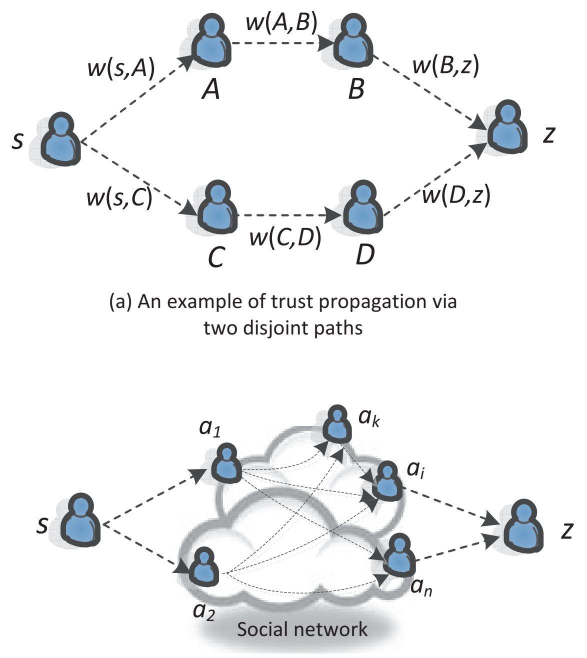
$$R_{s,z}(i) = \frac{g_{s,z}(i)}{\tau_{s,z}(i)}. \tag{4}$$

- Trust Score:

  From Eq. (3) and (4), s computes its trust score on z over N time windows as:

$$\begin{aligned} w(s,z) &= \beta \sum_{i=1}^{N} \alpha^{-i} \left[\lambda \sum_{t=1}^{m} a_{s,z}^{(t)} \frac{n_{s,z}^{(t)}(i)}{\Delta t_{s,z}^{(t)}(i)} \right. \\ &+ \left. (1-\lambda) \frac{g_{s,z}(i)}{\tau_{s,z}(i)} \right], \end{aligned} \tag{5}$$

where α is the discount factor (e.g., $\alpha = 2$) to count the latest experience more heavily than the past ones, λ is the trade-off value to adjust how much to put weight on social trust and service trust, and β is the normalization factor ensuring $w(s,z) \in [0,1]$. The graphical representation of the trust score computation from s on z is illustrated in Figure 14(b).



(a) An example of trust propagation via two disjoint paths

(b) Trust propagation in real social network

Figure 15. Graphical representation of trust score of two non-neighbors.

3 Trust Propagation in Social Networks

We now show how to compute the trust score of node s on node z where they are not neighbors. Computing the trust score between two non-neighbors in the social network is challenging because of the sophisticated social connections between them. In addition, there are possibly several paths with different lengths that the source node reaches to the target storage node. Consider a simple example in Figure 15(a) where source node s reaches to storage node z via two social paths with four intermediate nodes A, B, C, and D. From the previous subsection and Eq. (5), node s can compute the trust scores on its neighbors A and C as $w(s,A)$ and $w(s,C)$. However, it cannot compute its trust score on z as they are not neighbors. To tackle this issue, we adopt the approaches in [17] [18], which have been proposed to compute the trust propagation in web-based social networks. The main idea is to compute the trust score based on the referral scores of the intermediate nodes. For example, in Figure 15(a), s estimates its trust score on z by simply averaging the trust scores aggregated from the two social paths $s \to A \to B \to z$ and $s \to C \to D \to z$ with equally weighted factors:

$$\begin{aligned} w(s,z) = \frac{1}{2}[&w(s,A)w(A,B)w(B,z) \\ &+ w(s,C)w(C,D)w(D,z)]. \end{aligned} \tag{6}$$

In general, however, s reaches to z through a social network with sophisticated social connections shown in Figure 15(b). The interconnections between s and z typically are created in large size mesh networks. However, if s has complete global knowledge of the interconnections reaching to z, it can compute the expected trust score on z as:

$$w(s,z) = \Phi_i\left(\theta_{s,a_i}, w(a_i,z)\right), \tag{7}$$

where θ_{s,a_i} is the weight factor assigned by s over intermediate node a_i's trust score on z, and Φ denotes the aggregation function that combines the weighted trust scores.

Unfortunately, in real social networks, obtaining the complete global knowledge of the connections between two non-neighboring nodes is time consuming and sophisticated due to the large size of the network. In fact, this is an NP-Complete problem [19] [20]. To tackle this, the proposed SOS system searches for only the set of K best paths to the potential storage node. Particularly, suppose there are n nodes, including the source and the storage nodes, on the path from source s to storage node z, and $\mathbf{K}$ is the set of best K disjoint paths, the aggregated trust score of s on z can be mathematically written as:

$$w(s,z) = \frac{1}{K}\sum_{j\in\mathbf{K}}\prod_{[a_i,a_{i+1}]\in p_j(s,a_1,a_2,\ldots,a_{n-2},z)} w_j(a_i,a_{i+1}), \tag{8}$$

where $p_j(s,a_1,a_2,...,a_{n-2},z)$ is the jth path from s to z, $w_j(a_i,a_{i+1})$ is the weight factor assigned by a_i over a_{i+1}'s trust score on z. The question is how to find the best

K paths from a source node to a target node? Unfortunately, finding K best paths is also an NP-complete problem [43]. Therefore, to tackle this issue, we propose an approach which utilizes the Dijkstra's algorithm to search for the K best disjoint paths between the source and target nodes. However, the Dijkstra's algorithm searches for the shortest path while our objective is to find the paths with maximum trust values. Fortunately, this issue can be resolved by subtracting the trust value of each link from the unity trust value, i.e., the maximum possible trust value. For example, the trust value between two node u and v is $w(u, v)$ which is computed by Eq. (4). In order to utilize the Dijkstra's algorithm we use the weight of the link between u and v as $1 - w(u, v)$. As a result, finding the shortest path is equivalent to finding the path with maximum trust value. Once a path has been obtained, we then remove the intermediate nodes on the identified path from the graph, and find the next shortest path using the same algorithm. The algorithm is repeated until K paths are identified or there are no more paths left. The algorithm is summarized in Algorithm 1.

Algorithm 1 Finding K Paths using modified Dijkstra's algorithm

```
1: Input($M, W, v_s, v_t, K$) {$M$ is an adjacency matrix that represents the sub network
   between $v_s$ and $v_t$. $W$ is the edge weight matrix. The edge weight $w_{a_{i,j}} =
   1 - Trust_{a_{i,j}}$ }
2: Output($(({p^1}_{v_s \to v_t}), ...({p^G}_{v_s \to v_t}))$) {${p^i}_{v_s \to v_t}$ is the $i^{th}$ shortest path found. $G$ is the
   number of feasible solutions. $G \leq K$ }
3: Dijkstra's Algorithm ($M, W, v_s, v_t$)
4: if Shortest Path Found then
5:     Add $p_{v_s \to v_t}$ to output paths.
6:     Remove intermediary nodes $p_{v_s \to v_t}$ from $M$.
7:     if $K$ paths Found then
8:         Return $K$ Paths
9:     else
10:        Goto Line 3
11:    end if
12: else
13:    return $G$ Paths {$G < K$}
14: end if
```

The time complexity of the algorithm is $O(K(m + nlog(n)))$ as we repeat the

Dijkstra's algorithm K times. Here m is the number of nodes and n is the number of social links. In the next section, we will evaluate the performance of the proposed SOS via analysis and simulations. In the next section, we will evaluate the performance of the proposed SOS via analysis and simulations.

F Fault Recovery

Whenever a node fails, data on that node must be recovered and must be written to a new node. Figure 16 shows a failure at node D. Data from node D must be recovered and written to Node F. Since we cannot access the failed node we can recover data from the existing nodes using either minimal storage technique. It can be used for fragment self-regeneration.

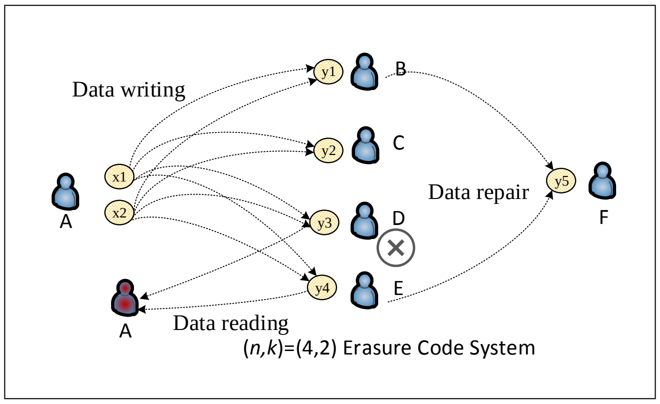


Figure 16. Node Failure

1 Minimal Storage Nodes (MSN)

Self-regeneration with minimal storage nodes is developed based on the traditional erasure code that allows the original file to be recovered by using any k out of n fragments. In the SDSF, each storage node has l fragments to achieve minimal storage nodes. Therefore, using any $l.k$ fragments from k node is able to regenerate new $l.k$ fragments in the new node. Figure 17 depicts the process that continues the example in Figure16. Suppose node is unreachable as shown in Figure 17. By connecting to any of two available nodes (e.g., v_2 and v_4 in Figure 17), the new node (e.g., v_5) can reconstruct the whole file and generate new encoded fragments (e.g., $c_{1,3}$ and $c_{2,3}$ in Figure 17). It is noted that $c_{1,3}$ in v_3 may not equal to $c_{1,3}$ in v_5, due to different coefficients. This approach repairs the fragments by minimal number of nodes and thus it is highly resilient for up to $((n-k))$ compromised nodes. In addition, it has the minimal number of connections (i.e., k) from the new node to storage nodes. The shortcoming of this approach is that the communication cost may not be minimized. The end-to-end communication overhead is $k \cdot l \cdot |c_{i,j}|$, where $|c_{i,j}|$ is the size of a fragment.

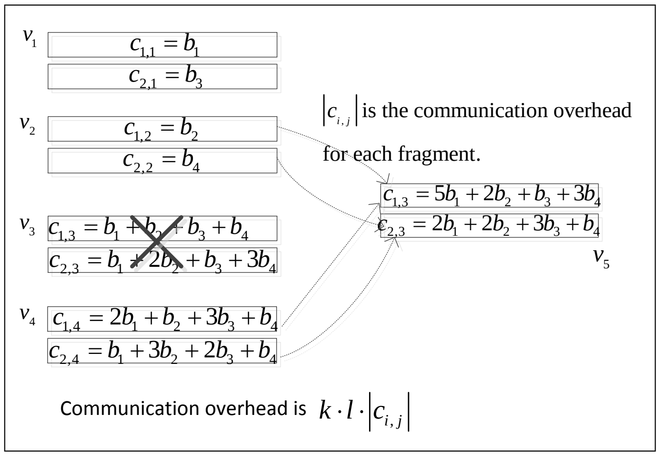

v_1
$c_{1,1} = b_1$
$c_{2,1} = b_3$
v_2
$c_{1,2} = b_2$
$c_{2,2} = b_4$
v_3
$c_{1,3} = b_1 + b_2 + b_3 + b_4$
$c_{2,3} = b_1 + 2b_2 + b_3 + 3b_4$
v_4
$c_{1,4} = 2b_1 + b_2 + 3b_3 + b_4$
$c_{2,4} = b_1 + 3b_2 + 2b_3 + b_4$
$|c_{i,j}|$ is the communication overhead for each fragment.
$c_{1,3} = 5b_1 + 2b_2 + b_3 + 3b_4$
$c_{2,3} = 2b_1 + 2b_2 + 3b_3 + b_4$
v_5
Communication overhead is $k \cdot l \cdot |c_{i,j}|$

Figure 17. Self-regeneration with minimal storage nodes

2 System reliability

For simplicity, suppose the storage devices fail following the same distribution with the probability of failure λ. The failure could be due to physical damage or security compromise. We assume that the failures of the devices are independent. In our proposed system, we implement the (n, k) coding scheme where the data is stored into n nodes and any k of them is sufficient to recover the original data. In order for the system to fail, there must be more than $n - k$ failures at the same time. Let X be the random variable denoting the number of storage devices which fail at a time instance. The probability that the system fails can be expressed as:

$$\begin{aligned}\Pr(\text{system fails}) &= \Pr(X > n-k) \\ &= 1 - \sum_{i=0}^{n-k} \binom{n}{i} \lambda^i (1-\lambda)^{n-i}. \end{aligned} \tag{9}$$

From Eq. (9), it is clear that when k increases, the system reliability decreases as it can tolerate less number of failures. In this case, the data redundancy that is necessary in the storage is reduced.

3 Data regeneration overhead

We now compute an average of how much data needs to be transferred over the network for data regeneration. Similarly, we assume that at any time instance, a data storage device fails with probability of λ. Let X be the random variable denoting the number of failed storage devices. We then can write the probability that i storage devices fail at the same time is

$$\Pr(X = i) = \binom{n}{i} \lambda^i (1-\lambda)^{n-i}. \tag{10}$$

In the case of MSN, whenever a storage device fails, it requires a transmission bandwidth of the original data piece, i.e., $f = m/l$, which is downloaded from the k storage devices. Let Y be a random variable denoting the amount of transmission bandwidth used for data regeneration. Therefore, we can compute the expected transmission bandwidth for data regeneration as:

$$\begin{aligned} E[Y] &= \sum_{i=1}^{n-k} i \times f \times \Pr(X = i) \\ &= f \sum_{i=1}^{n-k} i \binom{n}{i} \lambda^i (1-\lambda)^{n-i} \\ &= f \times \left(n\lambda - \sum_{i=n-k+1}^{n} i \binom{n}{i} \lambda^i (1-\lambda)^{n-i} \right), \end{aligned} \tag{11}$$

where the last equation is achieved by using $\sum_{i=0}^{n} i \binom{n}{i} \lambda^i (1-\lambda)^{n-i} = n\lambda$, i.e., the expectation of the Binomial distribution with parameters n and λ.

G Security

SDSF system has to protect the data on the individual unreliable data nodes. It has to verify the mobile node to verify that it is who it claims it is. SDSF system also has to make sure it has the authority to request the data. We have implemented security at various levels in our storage system. We use authentication, authorization, encryption and erasure coding for securing our data. We use AES encryption to encrypt the data files and use Shamir's secret key algorithm (explained below) to split the key into n keys. We store each key along with the n chunks generated from the erasure coding algorithm. In order for someone to illegally reconstruct the files they have to gain access to at least k servers. Unless they get access to the k servers, they cannot get the k keys to reconstruct the original key. Once they get the original key, only then will they be able to decrypt the k files. Also they need the k files along with the reconstructed key to decode the original file.

1 Shamir's Secret Key

We implemented Shamirś [44] secret key sharing algorithm to minimize the risk of someone getting access to the key. We will take an encryption key and will split the encryption key into n keys and store each key along with the n data nodes. The property of Shamir's secret key is, it needs k keys out of the n keys to retrieve the original Key K. Once we split the key into n sub keys we delete the original key. So in order to decrypt the file, we need to get the original key which can only be obtained by getting k sub keys. In order to get k sub keys, an intruder must get access to the k storage nodes. Suppose that our secret key is 7772. Consider $(n, k) = (6, 3)$. Choose $k - 1$ random keys. For example 55, 70. The polynomial function is $f(x) = 7772 + 55x + 70x^2$. We can construct 6 sub keys from the polynomial function. $(f(1), 7897), (f(2), 8162), (f(3), 8567), (f(4), 9112), (f(5), 9797), (f(6), 10622)$.

In order to reconstruct the original key we need any 3 sub keys. Consider we have f(2),f(4),f(5). $(x_0,y_0),(x_1,y_1),(x_2,y_2) = (2,8162),(4,9112),(5,9797)$ We compute Lagrange basis polynomials:

$$f(x) = 7772 + 55x + 70x^2$$
$$l_0 = \frac{x-x_1}{x_0-x_1}.\frac{x-x_2}{x_0-x_2} = \frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}$$
$$l_1 = \frac{x-x_0}{x_1-x_0}.\frac{x-x_2}{x_1-x_2} = -\frac{1}{2}x^2 + \frac{7}{2}x - 5$$
$$l_2 = \frac{x-x_0}{x_2-x_0}.\frac{x-x_1}{x_2-x_1} = \frac{1}{3}x^2 - 2x + \frac{8}{3}$$
$$f(x) = \sum_{j=0}^{k-1} y_j.l_j(x)$$
$$f(x) = 8162.(\tfrac{1}{6}x^2 - \tfrac{3}{2}x + \tfrac{10}{3}) + 9112.(-\tfrac{1}{2}x^2 + \tfrac{7}{2}x - 5) + 9797.(\tfrac{1}{3}x^2 - 2x + \tfrac{8}{3})$$
$$f(x) = 7772 + 55.x + 70x^2$$

So we recovered the polynomial and the secret key 7772.

H Data Management

Users must have security and reliability while storing files without sacrificing convenience and usability. Some of the files of a user are meant to be secure and the access should be limited to only the file owner and with whom ever the file owner has shared it with. But some files are meant for public (could be public for friends and acquaintances only or public for everyone) use and other users must be able to search and access them. If a file is marked as public by the user, though it is not encrypted, it is stored in distributed devices. As a result of storing it in distributed devices we cannot directly use the windows search [45] as it is used for searching for files stored on a single machine. We cannot use the Google search engine [46] or Bing search engine [47] for 2 main reasons:

1. Distributed storage:
   Using the Google and Bing search engines directly on the distributed files will rank them differently as compared to ranking them together as a single file. Also the meaning from the split files could be interpreted differently as compared to the combined file.

2. No Meta tags:

   Google search engine or any other search engine mainly relies on the website developers to implement the Search Engine Optimization (SEO). SEO teaches Google about the site so it can rank accordingly. Some of the SEO techniques include specifying title, keywords, meta tags, schema. The developer of the website specifies these details which will be used along with the content of the website to rank the websites. However, in our storage system, the file owner will not have to provide the tags for the documents which will make it difficult to search for the documents efficiently.

Information Fusion in distributed systems Chapter IV will explain how to search for publicly shared files efficiently in our social distributed storage system.

CHAPTER IV

INFORMATION FUSION IN DISTRIBUTED SYSTEMS

Approaches such as mediation data fusion [48] as shown in Figure 18 perform the distributed fusion in order to avoid the collection of data into a centralized server. In the mediation information fusion, *Mediator* transforms a user query into several

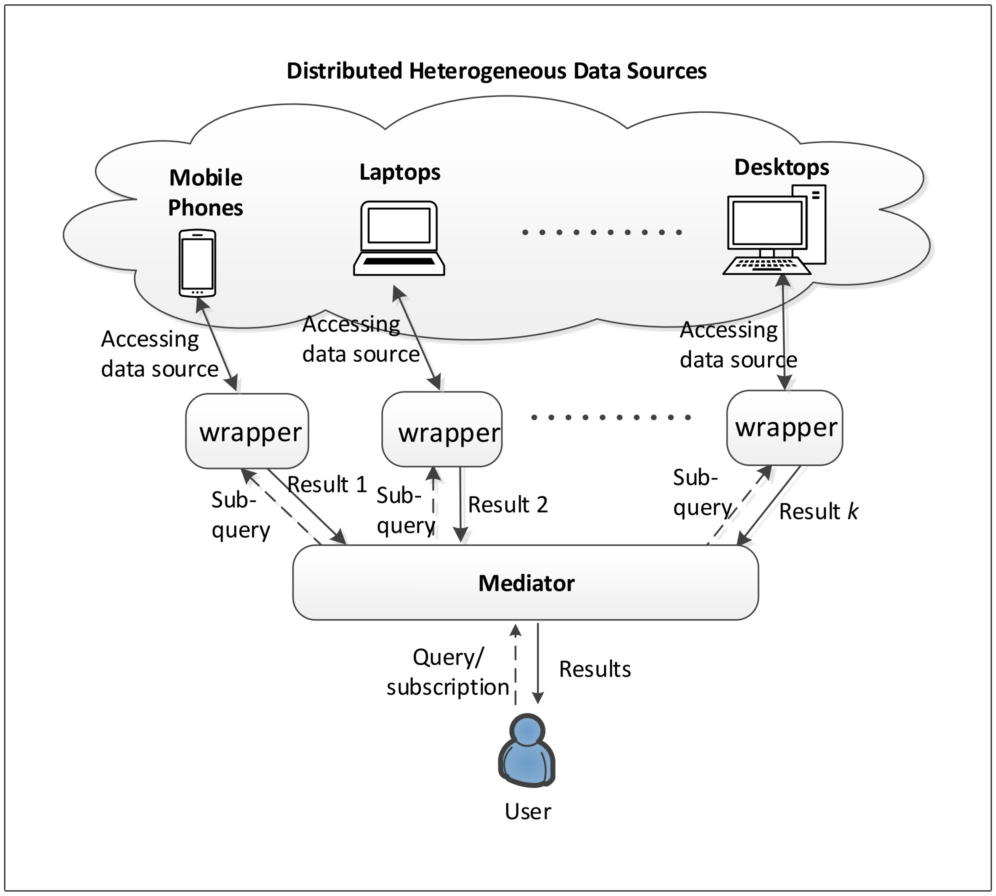


Figure 18. Multi-intelligent Information Fusion Approaches

sub-queries which are further transmitted to the data sources where they are executed.

Wrappers then extract the data knowledge from the local data sources. Figure 18 shows the process in which a user's query leads to data fusion that is performed in a distributed manner. Mediator breaks the query into sub-queries and gives them to the Wrapper. Wrappers separately access the distributed and heterogeneous data sources (e.g., mobiles, laptops, desktops and servers) as shown in Figure 18. In the end, the results from local wrappers will be integrated at the Mediator before displaying them to the user. The mediation information fusion is well scalable even for a large number of data sources.

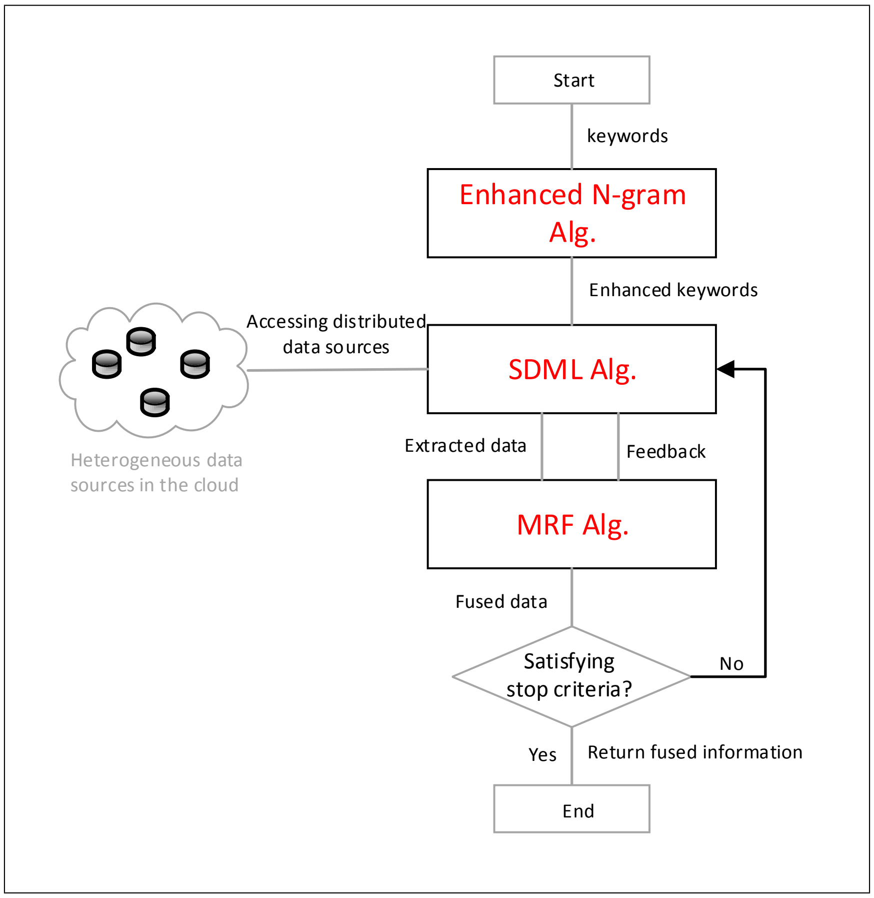


Figure 19. NSM Information Fusion Procedure

We perform information fusion on the cloud using N-gram, Semantic Machine Learning, and Map Reduce (NSM) Algorithms. The innovations of the proposed NSM is to improve the information fusion accuracy and efficiency for processing heterogeneous, distributed, and massive data on the cloud for tactical information fusion services. The information fusion could start with a query request issued by a user, e.g., from a mobile device. The query consists of various fusion parameters like keywords, geographic information, time and other contextual information, and they will be used for further filtering and fusion. Once the query is submitted to the fusion engine, NSM then performs information fusion as shown in Figure 19 to find relevant information.

A Technical Innovations:

While the details of the proposed NSM algorithms will be described in the section E, the significant innovations of NSM are summarized as follows:

- Enhanced N-gram Algorithm: Our enhanced N-gram algorithm understands the fuzzy query on two levels. In the first level, Enhanced N-gram Algorithm has the ability to correct the misspelled query parameters. At the higher level, it breaks a query into precise tokens to understand the needs of the user by qualifying the contextual association.
- SDML Algorithm:
  The proposed SDML algorithm allows the machine learning enabled information extraction and semantic mappings. This property is highly desirable in the tactical environments, where the information comes from different sources across the network nodes.
- MRF Algorithm:
  The proposed MRF algorithm provides an efficient way to fuse information retrieved from multiple data sources. Importantly, the trustworthiness of the information sources is also modeled and taken into account when performing reasoning and integration.

B N-grams Overview

N-gram are used to process ambiguous user queries. N-grams are explained in great detail in [49]. N-grams can predict the next word from the previously $N-1$ words. Assume we are trying to find the probability of a word "vehicle" given the occurence of the words "The thief stole a" i.e. we are trying to find $P(vehicle \mid The\ thief\ stole\ a)$. We can use conditional probability.

$$P(vehicle|The\ thief\ stole\ a) = \frac{Count\ of(The\ thief\ stole\ a\ vehicle)}{Count\ of(The\ thief\ stole\ a)} \tag{12}$$

$P(w_1, w_2, ...w_n)$ denoted as $P(w_1^n)$ is the probability of the n words appearing in that order. Chain rule of probability can be applied to solve this. Chain rule of probability states that:

$$P(X_1...X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1^2)...P(X_n|X_1^{n-1}) \tag{13}$$

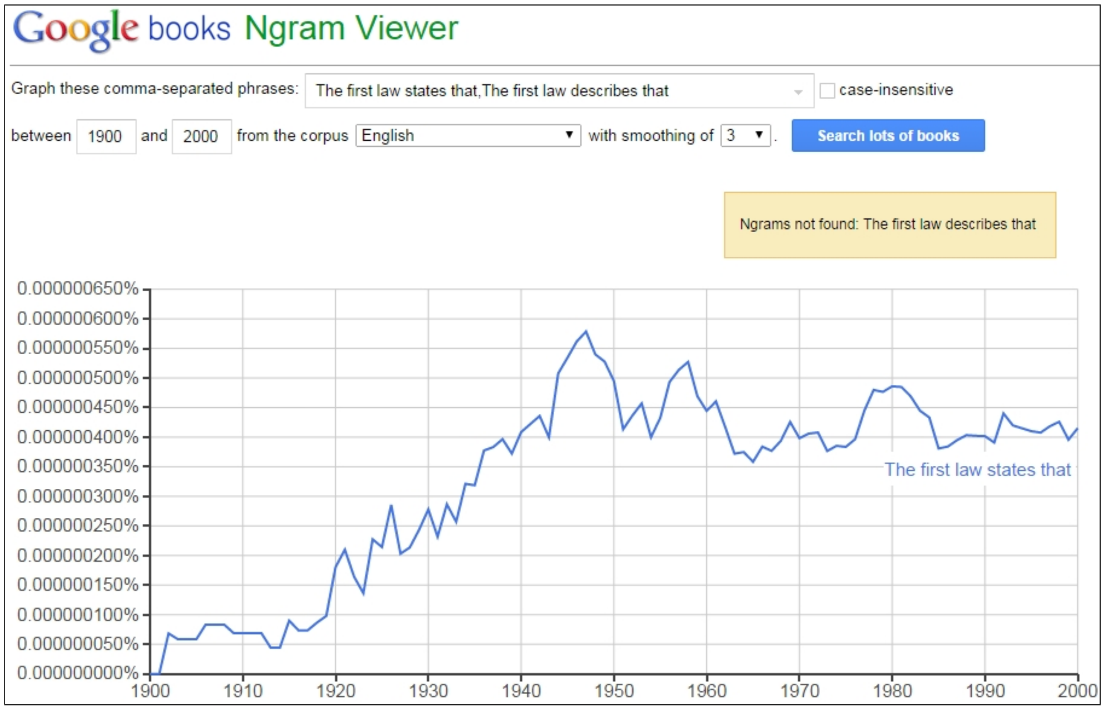
$$P(X_1...X_n) = \prod_{k=1}^{n} P(X_k|X_1^{k-1}) \tag{14}$$

From Eq. (13) and Eq. (14) we can apply chain rule of probability to get the probability for n words to be in a given sequence:

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)...P(w_n|w_1^{n-1}) \tag{15}$$

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \tag{16}$$

Figure 20 shows the Google N-gram viewer for books. Suppose a user is trying to search for Newton's first law of motion. Assume that the user types in "The first law describes that". If we are trying to correct the query, and if our alternative is "The first law describes that", and if we perform N-gram on both the queries the former query does not even show up while the latter query has a probability. This way we can replace the former query with the corrected query.


Graph these comma-separated phrases: The first law states that,The first law describes that

case-insensitive

between 1900 and 2000 from the corpus English with smoothing of 3.

Ngrams not found: The first law describes that

Figure 20. Sample Google N-Gram Viewer

C Keyword Indexer

In reality it is not feasible to analyze all the documents available on all the storage devices as we do it in real time and the results have to be given to the user immediately. So we get the keywords and classify the documents before performing the erasure coding and storing them in distributed storage devices. Once we get the keywords we perform the following:

- Remove stopwords: We remove all the stop words like ‘a’, ‘the’ etc.
- Normalization: We perform normalization by removing period symbols, converting all characters to lower cases etc.
- Stemming: We then perform stemming by removing the ‘ing’, ‘s’ etc.
- Assigning weights: We then assign weights to the documents pointing by these keywords based on the word count, rarity of keyword etc.

We then store these keywords in indexing servers. Figure 21 shows the process of classification and file retrieval. When a user uploads a file, we extract the keywords and store them in indexing servers along with the file ID, location and synonyms. When a user is searching for files, we get only the related files from the indexing servers and perform NSM only on those files. This way we don't have to perform fusion on millions of documents, but we can perform only on the relevant documents saving both time and computation power.

D Our Approach to Information Fusion

- Enhanced N-gram Algorithm:
  Our enhanced N-gram algorithm understands the user query and contextual factors. It clarifies the query request on two levels:
  - Word Level: N-gram algorithm corrects the misspelled keywords.
  - Query Level: N-gram algorithm associates the request with the contextual knowledge to generate a more precise query.

- Semantic and Distributed Machine Learning (SDML) Algorithm:
  SDML performs semantic data mining in a distributed way across heterogeneous data sources. Our wrappers optimize the learning process and our mappers semantically translate the data back to the targets.

- Map Reduce-based Fusion (MRF) Algorithm:
  After obtaining the results from the SDML mining, MRF performs MapReduce-based fusion. It promotes in-depth information fusion and collaboration. The results obtained from a data source can be used as a new clue to learn other data sources and retrieve relevant information. MRF returns the results to the user only if the stop criteria is satisfied, as shown in Figure 19. Otherwise, MRF iteratively exploits collaborative learning and mining until the result is optimal.

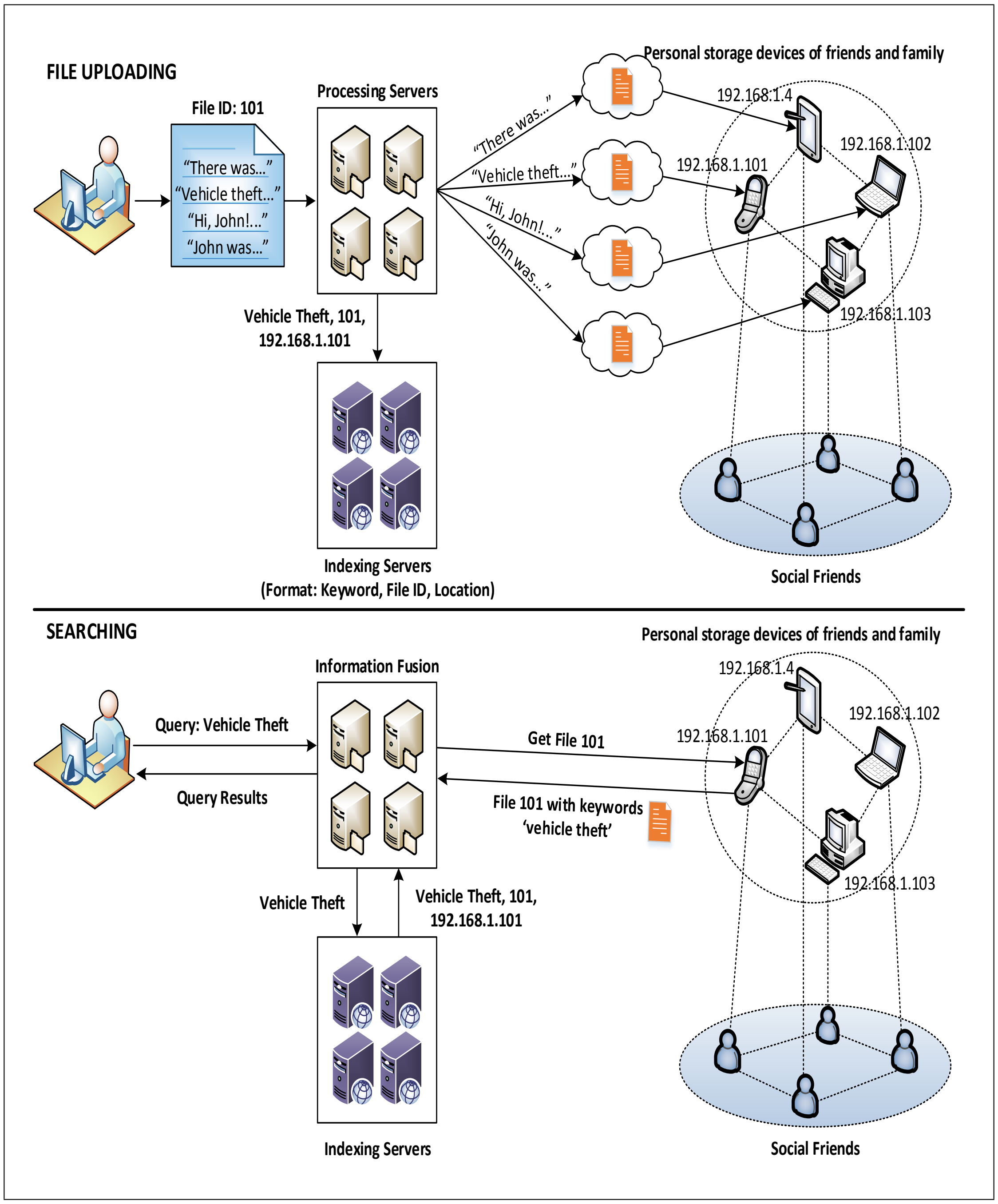


Figure 21. Keyword Extractor Prior to distribution

NSM is specifically developed for real-time information fusion with high precision and low runtime. It provides a fuzzy-based query that understands the user's need in a given context. NSM uses distributed machine learning to discover the useful information from unstructured data across a number of heterogeneous data sources. More importantly, it provides performance benefits to satisfy the information fusion services:

- Fusion Precision: NSM improves the fusion precision across heterogeneous data sources in the following way:
  - Query Accuracy: Our enhanced N-gram algorithm uses the contextual information and external knowledge to generate an accurate query that leads to a precise outcome from the query.
  - In-depth Collaboration: SDML improves the accuracy by an iterative processing across clouds. It allows the information obtained from a data source to be used as new input feedback to other data sources in order to find more relevant information.
  - Semantic Integration: MRF designs the semantic mapping function to improve the accuracy by allowing the information from different sources to be contextually integrated.
- Real-time Capability: NSM is adapted to distributed environment and our algorithms are optimized to satisfy the real-time tactical applications.

E Enhanced N-Gram Algorithm

N-gram [50] estimates the Edit Distance for detecting and correcting typos in a query. We propose the enhanced N-gram algorithm to understand the entire query, instead of individual keywords only. It uses an N-Gram-based probabilistic model to precisely refresh the user's request for a fuzzy query.

Enhanced N-gram algorithm takes advantage of two external resources that are available:

- Contextual Information:

  The contextual and situational knowledge of the user can be used to better understand the user query. The knowledge could be the time, geographic information, age etc. that allows us to contextually filter content for a specific area and time of interest such that the specific contextual questions are addressed in the distributed learning and fusion.

- Lexical resource:

  Lexical resource is a collection of user-defined resources and the open resources available on the internet like Wikipedia and WordNet. The use of these resources provide linguistic information about words and relationship between concepts of natural languages.

Let w be a keyword and $\mathbf{D}$ be the collection of all available contextual and lexical resources. Given a fuzzy-based query, the enhanced N-gram algorithm performs (i) Correcting typos in a query and (ii) Understanding the entire query, as illustrated below.
Correcting typos in a query: It corrects the typos in a query which consists of a set of words w. Let $\mathbf{U_D} \subseteq \mathbf{\Sigma}^*$ be a finite universal lexicon, which conceptually consists of all the words in $\mathbf{D}$. We define the set $\mathcal{C}_{\mathbf{D}}(w)$ of candidates as:

$$\mathcal{C}_{\mathbf{D}}(w) = \{w\} \bigcup \{w' \in \mathbf{U_D} | ed(w, w') \leq \delta\}, \tag{17}$$

where δ is a pre-specified threshold and $ed(w, w')$ is the *edit distance* between two words w' and w. In our framework, the edit distance $ed(w', w)$ is defined as the number of *insertion, deletion, transpose* and *substitution* operations to transform w' to w.

- N-grams based similarity scores:

To identify the candidates of the input query, we use the *similarity score* function over $\mathbf{D}$. Particularly, the similarity score between two strings w' and w using the $N-grams$ model is expressed as:

$$score_{\mathbf{D}}(w', w) = \frac{|w' \bigcap w|}{|w' \bigcup w|}, \tag{18}$$

where $|w' \bigcap w|$ and $|w' \bigcup w|$, respectively, denote the number of similar $N-grams$ in w' and w and the number of unique $N-grams$ in the union of w' and w. For example, the similarity score of the misspelled word w' = "vahicl" and the correct word w = "vehicle" using an $N-grams$ with $N = 2$, i.e., bi-gram, is 3/8. This is because using bi-grams model, w' is represented as $\{``va", ``ah", ``hi", ``ic", ``cl"\}$, whereas w is represented as

$$\{``ve", ``eh", ``hi", ``ic", ``cl", ``le"\}.$$

Therefore, the number of similar bi-grams $w' \bigcap w = 3$, i.e., {"hi","ic","cl"}, and the total number of unique bi-grams in the union $w' \bigcup w = 8$, is

$$\{``ve", ``eh", ``hi", ``ic", ``cl", ``le", ``va", ``ah"\}.$$

- Selecting candidate set:

  We denote the candidate set as $top_{\mathbf{D}}(w)$ which is a set of k words $w' \in \mathcal{C}_{\mathbf{D}}(w)$ with the highest k $score_{\mathbf{D}}(w', w)$. If $|\mathcal{C}_{\mathbf{D}}(w)| < k$, then $top_{\mathbf{D}}(w)$ is simply $\mathcal{C}_{\mathbf{D}}(w)$.

Understand the Entire Query: The input query usually consists of several keywords with contextual association. It is not enough for just correcting individual keywords without considering query semantic and context. In other words, we need to understand the entire query in the given contextual environment. We assume that the input query, possibly consisting of misspelled words, $\mathbf{s} = \langle s_1, s_2, \ldots, s_n \rangle$. As shown above, i.e., the word-level correction, the algorithm creates a set $\mathcal{C}_{\mathbf{D}}(\mathbf{s})$ of suggestions and determines the values $score_{\mathbf{D}}(\mathbf{r}, \mathbf{s})$ for $\forall \mathbf{r} \in \mathcal{C}_{\mathbf{D}}(\mathbf{s})$. In addition, for a word $w = w_1 w_2 \cdots w_m$ of m characters, we denote by $w_{[i,j)}$ the word $w_i w_{i+1} \cdots w_{j-1}$. Further, we denote the word $w_i w_k$ as a concatenation of w_i and w_k. Furthermore, we denote $\lfloor \mathbf{s} \rfloor = s_1 s_2 \cdots s_m$, a single word, which is constructed by concatenating the words of $\mathbf{s}$. For instance, if $w_1 =$ "vehicle" and $w_2 =$ "theft", then $\mathbf{s}$ corresponds to the query "vehicle theft", whereas $\lfloor \mathbf{s} \rfloor$ is the word "vehicletheft", and $\lfloor \mathbf{s} \rfloor_{[1,8)} =$ "vehicle".

- Query segmentation:

  Query segmentation plays an important role in breaking the input query into precise tokens. For example, if the user enters "vahiclethef", a good system should identify it as a misspelling and correct it to "vehicle" and "theft" as tokens, as those are in fact the tokens that the user had in mind.

  - Token candidate:

    We define $w = \lfloor \mathbf{s} \rfloor_{[i,j)}$ a *token candidate* if w is a word of $\mathbf{s}$, or there is a word $w' \in \mathcal{C}_{\mathbf{D}}(\mathbf{s})$ such that $score_{\mathbf{D}}(w', w) > \epsilon$ for some pre-specified value ϵ. Intuitively, w is a token candidate if it is an original token of $\mathbf{s}$, or if we have a high confidence in our word-level suggestion to correct w.

  - Tokenization:

    We define a *tokenization* of $\mathbf{s}$ as a sequence $\mathbf{J} = \langle j_1, j_2, \ldots, j_l \rangle$ with $j_1 = 1$ and $j_l = m + 1$. Then the tokenization $\mathbf{J}$ induces a set of tokens $\langle \lfloor \mathbf{s} \rfloor_{[j_1,j_2)}, \lfloor \mathbf{s} \rfloor_{[j_1,j_2)}, \ldots, \lfloor \mathbf{s} \rfloor_{[j_{l-1},j_l)} \rangle$. $\mathbf{J}$ is called a *tokenization candidate* if each token induced by $\mathbf{J}$ is a token candidate.

- Query correction optimization:

  The query correction is formulated as an optimization problem to find the optimal tokenization $\mathbf{J}^*$ of the input query:

$$\mathbf{J}^* = \underset{\mathbf{J}}{\operatorname{argmax}} \left[\sum_{i=1}^{l} score_{\mathbf{D}}(w_i', w_i) \right], \tag{19}$$

  where $w_i = \lfloor \mathbf{s} \rfloor_{[j_i,j_{i+1})}$ and $w_i' \in top_{\mathbf{D}}(w_i)$. The intuition of the optimization formulation expressed in Equation (19) is that the algorithm scans all the possible tokenization's of the input query and finds the best one which maximizes the summation of the similarity scores of all induced tokens. In practice, the algorithm runs very quick because it considers only misspelled tokens while ignoring the correctly spelled words. For example, if the user enters "vahiclthef near forward operating base in Delhi, Afghanistan", the algorithm will focus on correcting "vahiclthef" and ignore the other tokens as they are correctly identified in **D**.

The output of the QP algorithm is tokens that are sent to the distributed data mining agents which semantically reformulate the query in the local ontologies to retrieve the relevant information from the local data sources.

F SDML Algorithm

The SDML algorithm performs (i) Information extraction by selecting appropriate ontology to associate with the data sources, and (ii) Data mining by semantic mapping and auto-learning.

1 Information Extraction:

Each cloud node n_i extracts the useful information from its data source. The SDML algorithm, with little training, can self evolve and automatically identify entities, relations, and co-references from the data source, and has the following three components:

Topic Identifier:

We implement the clustering method based on a predefined set of most commonly used words to identify topics of the data source. In this method, data sampled from the data source (e.g., text documents) is represented as an unordered set of words, regardless of grammar. The frequency of each word occurrence is then used as a feature for training a classifier. For example, we have two text documents: "a car was stolen last week" and "a truck was stolen by a person". Here we use Map Reduce to count the number of keywords. Figure 22 shows an example of how word count is

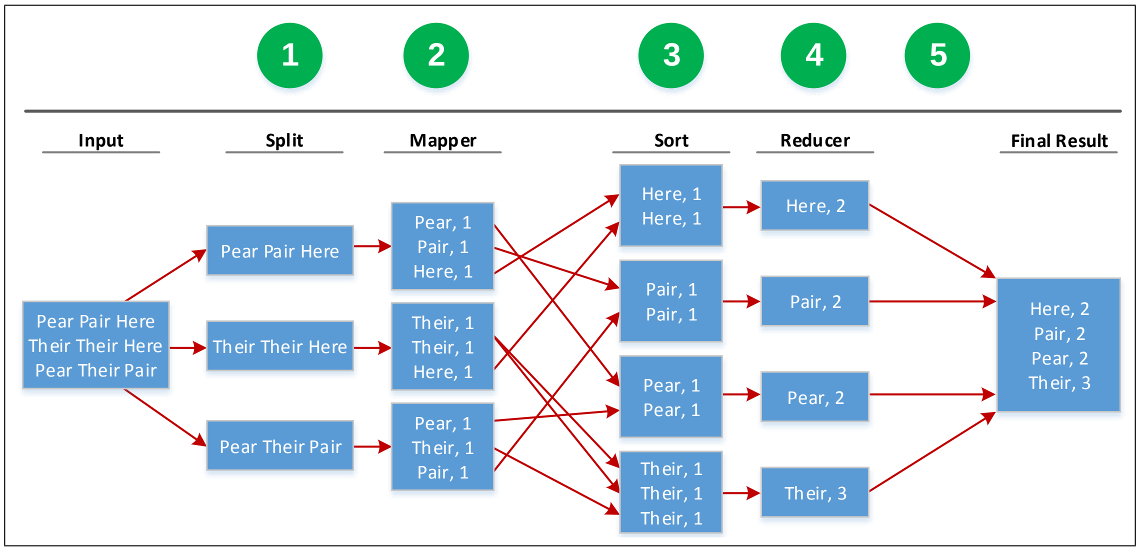


Figure 22. Map Reduce word count

done using Map Reduce. Based on the data source, a dictionary can be constructed as $\{$"a" : 1, "car" : 2, "$person$" : 3, "$truck$" : 4, "$stolen$" : 5, "by" : 6, "$last$" : 7, "$week$" : 8, "was" : 9$\}$, consisting of nine distinct words. The two documents are then represented as vectors, i.e., [1,1,0,0,1,0,1,1,1] and [2,0,1,1,1,1,0,0,1], where each entry of the vectors depicts the count of the corresponding entry in the dictionary (i.e., histogram representation). Based on the vector representation, the algorithm can quickly compute the occurrences of important words such as "stolen" (twice), "car" and "truck" (each once but related.) Based on the distribution of the words, it then classifies the documents into the *Vehicle-Looting* category. In addition, depending on the purpose and the required level of the classification, the system administrator can apply weight factors to each term based on pre-specified categories of interests. This can be achieved from "a study of term weighting schemes using class information for text classification" [51] with slight modifications. Based on the outcome result, the algorithm selects an appropriate ontology to attach to the data source. Correctly selecting the ontology is important as the data source can be precisely represented in a formal logical language. A parser such as the full Stanford parser can be used to parse the document before performing data mining.

Entity Extractor:

Entity extractor is a process for automatically extracting metadata information from the data source (e.g., unstructured text documents). This is a key step allowing *semantic* query and information integration from several data sources. The keyword or string-based approaches show several shortcomings and potentially miss critical information. For example, keyword-based query approaches will not be able to find documents published within a specific time window due to the inconsistencies of data formats in the documents. We leverage the powerful and open source architecture of Apache Unstructured Information Management Architecture (UIMA) [52] for processing unstructured information. More importantly, UIMA has a "plugin" architecture which allows several developers to create and extend components which are inter-operable. These plugins are called "annotators". We can use the builting annotators or we can write custom annotators. Some of the available built-in annotators [53] are:

- Whitespace Tokenizer Annotator:
  This is the most simplest annotator. It tokenizes the documents simply based on white spaces. This is not very useful.

- Snowball Annotator:
  The Snowball annotator uses the stemming algorithm and creates the features for each token.

- Regular Expression Annotator:
  The Regular Expression Annotator detects entities like email addresses, URLs, phone numbers, zip codes or any other entity based on regular expressions and concepts.

- Dictionary Annotator:

  The Dictionary Annotator creates annotations based on word lists that are compiled to simple dictionaries.

- BSF Annotator:

  BSF Annotator is used to create annotations for scripting languages that are supported by Apache BSF

- OpenCalais Annotator:

  OpenCalais can detect a large variety of entities, facts and events like for example Persons, Companies, Acquisitions, Mergers, etc.

- Concept Mapper Annotator:

  ConceptMapper is a powerful, highly configurable dictionary UIMA-based annotator. Numerous parameters can be used to specify various aspects of the lookup algorithm, input processing and output options.

- Configurable Feature Extractor Annotator:

  The Configurable Feature Extractor (CFE) Annotator is a multipurpose tool that enables feature extraction from a UIMA CAS in a very generalized and application independent way.

- Tika Annotator:

  Apache Tika is used for detecting and extracting metadata and structured text content from various documents using existing parser libraries.

- AlchemyAPI Annotator:

  The AlchemyAPI Annotator is a wrapper for the AlchemyAPI webservices which provide text enrichment facilities like categorization, entity extraction, language identification, keyword extraction, concept tagging etc.

- User Created Custom Annotator:

UIMA allows for custom creation of Annotators. We can specify what type of annotator we want.

After this process, type-specific annotations and metadata will be added to the unstructured document (i.e., document enrichment). For example, consider a document contains the following text: "On May 30th, 2013, Bob stole an SUV". If the document is run through the entity extraction algorithm, the XML-like annotations will be automatically added to the text and produce the following output: "On <entity date = "2013-05-30"> May 30th </entity><entity person="Bob">Bob</entity> stole a <entity vehicle="SUV"> SUV</entity>". As shown, the entity keyword is wrapped around three entities identified in the text. After this step, unstructured data is transformed into structured data allowing easy data query and analysis. We denote D_i which is the transformed data source associated with cloud node n_i.

Validation:

Machine learning approach allows for fast and efficient data identification and extraction. However, it may incorrectly identify entities or may not be able to determine critical entities due to the hidden semantics of the data source. Our proposed approach provides a "validation" step which allows the administrator to review the found solutions with low confidence. The confidence threshold is adaptively set based on the information source to minimize the missed detections during the learning stage.

2 Data Mining Function:

Data mining function consists of two components.

Wrappers

Wrapper r_i is a specific program associated with a data source that knows how to query and extract data from the source via the source's query interface. The wrapper is also designed to be able to translate between the source data (e.g., in HTML format) and the data that conforms to the source schema (e.g., a set of tuples). In particular, r_i interacts with D_i via the local interface to extract relevant information passed by the input query or subscription.

Semantic Mapping (SM)

The *SM* maps the attributes of the entity identified by the local IEs associated with the local ontologies to the attributes of the target ontology, creating a unified representation of the entity. The mapping function is performed based on the *semantics* of the attributes, instead of a *keyword-based* strategy as implemented in the existing approaches. For example, consider two concepts where "*Car*" is used in the local ontology and "*Vehicle*" in the target ontology. Using keyword-based matching approaches, the two concepts cannot be matched and will not be integrated. On the other hand, using *semantic-based mapping*, the two concepts will fall into the same category (i.e., a thing used for transporting people or goods on land); thus, they will be identified and integrated during the fusion process. It boils down to how to semantically match the concepts between the sources' ontologies and the target ontology. To tackle this, we propose a probabilistic machine learning approach which automatically learns and maps between structured representation of data. The architecture of the SM is illustrated in Figure 23. It consists of three key modules: Distribution Estimator (DE), Similarity Estimator (SE), and Labeler.

- Distribution Estimator (DE):
  DE takes two ontologies O_l (i.e., local ontology) and O_t (i.e., target ontology) as its input, together with their data instance sets U_l and U_t, respectively. The data instances might be generated by the domain experts or sampled from the data sources. DE implements a machine learning technique to determine for every pair of concepts $\langle A \in O_l, B \in O_t \rangle$ their joint probability distribution (i.e., $P(A,B), P(\overline{A},B), P(A,\overline{B})$, and $P\overline{A},\overline{B})$. Additionally, we denote $|U_l|$ and $|U_l^{A,B}|$, respectively, the size of U_l and the number of instances in U_l that belong

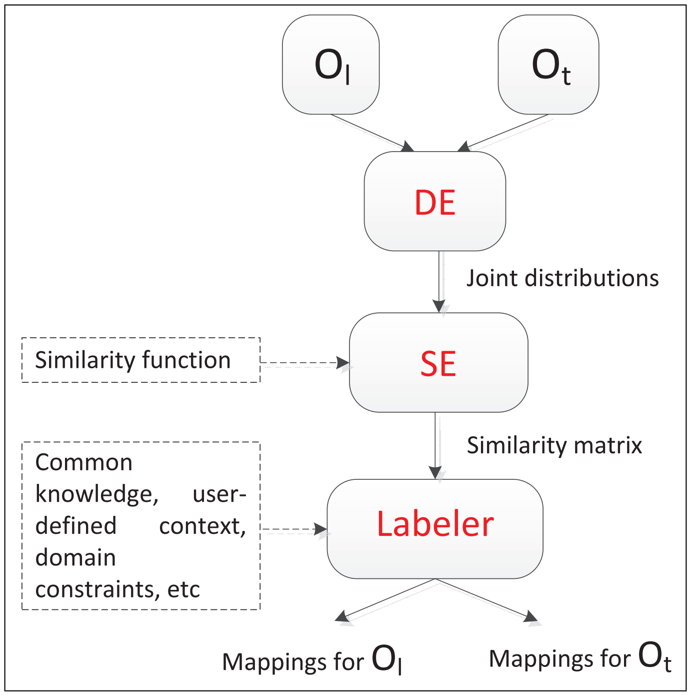


Figure 23. Semantic Mapping

to both A and B. DE performs the following steps:

- Partitioning Instances:

  Instances in the two ontologies are partitioned into two sets: U_l^A & $U_l^{\overline{A}}$ and U_t^B & $U_t^{\overline{B}}$, where U_i^X represents the set of instances that belongs to X and $U_i^{\overline{X}}$ represents the set of instances that do not belong to X.

- Training Learner:

  This step first trains a learner L for instances of A using U_l^A and $U_l^{\overline{A}}$ as the sets of positive and negative training examples, respectively. Such an example of learner training is shown in Figure 24 where $t1, t2, t3$, and $t4$ are instances of concept A, whereas $t5, t6$, and $t7$ are instances not belong to A.

  L is then applied to U_t^B to obtain two sets $U_t^{A,B}$ and $U_t^{\overline{A},B}$. Similarly, L is applied to $U_t^{\overline{B}}$ to obtain two sets $U_t^{A,\overline{B}}$ and $U_t^{\overline{A},\overline{B}}$. This process is illustrated in Figure 25. Applying the same learning process with the roles of O_l and

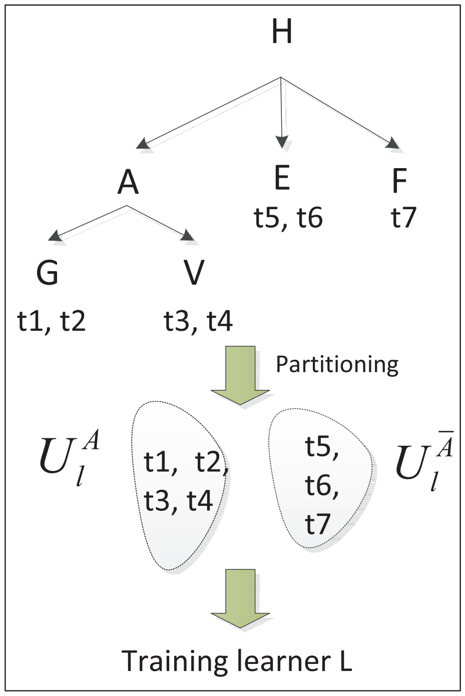


Figure 24. Training learner L for A

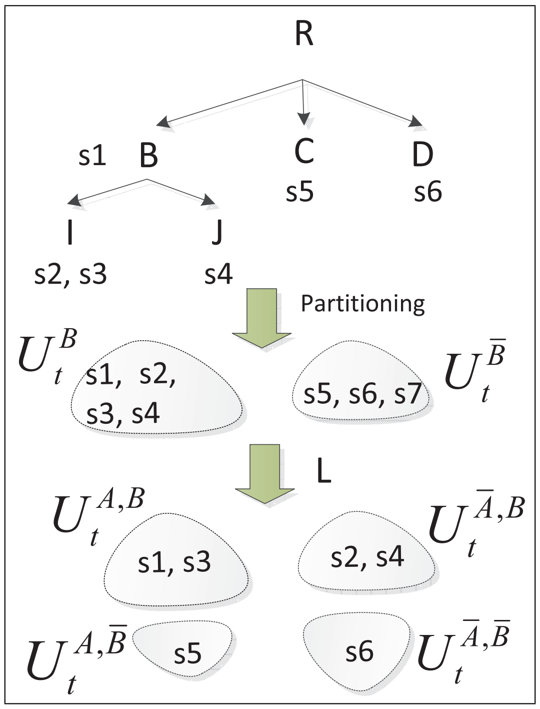


Figure 25. Applying learner L for B

O_t being reversed to obtain the sets $U_l^{A,B}, U_l^{\overline{A},B}, U_l^{A,\overline{B}}$, and $O_l^{\overline{A},\overline{B}}$.

In this approach, each learner L_i will learn a certain type of information from the training system and will be combined at the meta-learner.

– Estimating Joint Distribution:

When the partitioning and training processes are complete, DE then computes the joint distributions of the concepts. For example, the joint probability $P(A, B)$ is computed as the ratio between the total number of instances belong to both A and B in both ontologies O_1 and O_2 to the total number of instances:

$$P(A, B) = \frac{|U_l^{A,B}| + |U_t^{A,B}|}{|U_l| + |U_t|}. \tag{20}$$

The other three joint probabilities are computed using the same equation (20), using the sets $U_l^{\overline{A},B}, \ldots, U_t^{\overline{A},\overline{B}}$ obtained by L.

We then can obtain all the desired joint distributions by applying the above procedure to all pairs of concepts $\langle A \in O_l, B \in O_t \rangle$. The DE can be implemented in parallel to speed up the mapping where each distributed node learns to map subset of concepts in the ontology.

- Similarity Estimator (SE):

  SE uses the Jaccard method to measure the similarity between each pair of concepts $\langle A \in O_l, B \in O_t \rangle$. Particularly, we define that

$$Jaccard - sim(A, B) = P(A \cap B)/P(A \cup B) = \frac{P(A, B)}{P(A, B) + P(A, \overline{B}) + P(\overline{A}, B)}. \tag{21}$$

  The $Jaccard - sim(A, B) \in [0, 1]$, and it takes the lowest value 0 when A and B are disjoint, the highest value 1 when A and B are the same concept. The output from this module is a *similarity matrix* between the concepts of the two ontologies.

- *Labeler*:

  The labeler takes the similarity matrix, the domain constraints, and user-defined knowledge and searches for optimal mapping configuration that best satisfies the domain constraint and the common knowledge. The domain constraints can be domain-independent constraints (e.g., union - if all children of concept X match concept Y, then X also matches Y), and domain-dependent constraint (e.g., subsumption - if Y is not a descendant of X, and Y matches "Car", then it is unlikely that X matches "Vehicle"). We utilize the *relaxation labeling* approach which has been successfully applied to similar matching problems in computer vision and natural language processing for matching concepts between two ontologies. For example, we want to find the mapping from O_l to O_t, it simply becomes how to find the best label assignment to concepts in O_l, given all knowledge about the domains and ontologies. Let X be a concept in O_l and $\mathcal{L}$ be a label (i.e., a concept in O_t). Further, we let δ_K represent the knowledge of the domain (e.g., tree structures of the ontologies, sets of instances, set of domain constraints, etc). Mathematically, the matching problem can be written by the following conditional probability:

$$P(X = \mathcal{L}|\delta_K) = \sum_{M_X} P(X = \mathcal{L}, M_X|\delta_K) = \sum_{M_X} P(X = \mathcal{L}|M_X, \delta_K)P(M_X|\delta_K), \tag{22}$$

  where the sum is over all possible label assignments M_X to concepts other than X in O_l. When the concepts' label assignments are conditional independent on δ_K, the last term in Equation (22) then can be further written as:

$$P(M_X|\delta_K) = \prod_{X_i=\mathcal{L}_i \in M_X} P(X_i = \mathcal{L}_i|\delta_K). \tag{23}$$

  Without loss of generality, we assume that concept X has m features, where each feature is represented by a function $f_i(M_X, \delta_K, X, \mathcal{L})$. It is challenging to perform the matching between two concepts consisting of several features. To tackle this issue, we use the *signoid* function $\sigma(x) = 1/(1+e^{-x})$, which has been used to combine multiple sources of evidence where x is a linear combination of the features f_i. The key idea of using the signoid function is that it is essentially a smoothed threshold function for combining evidence from the different features with different weights, i.e., α_i. Thus, the matching problem can be written as:

$$P(X = \mathcal{L}|\delta_K) \propto \sum_{M_X} \sigma \left(\sum_{i=1}^{m} \alpha_i f_i(M_X, \delta_K, X, \mathcal{L}) \right) \times \prod_{X_i = \mathcal{L}_i \in M_X} P(X_i = \mathcal{L}_i|\delta_K), \tag{24}$$

where $\propto$ denotes "proportional to" and the weight α_i represents the importance of the corresponding feature f_i. To determine two concepts are matched or not, we use a threshold-based approach. In particular, we denote θ_l and θ_h as threshold low and high, respectively. If $P(X = \mathcal{L}|\delta_K) > \theta_h$, the system determines that the two concepts are matched. On the other hand, if $P(X = \mathcal{L}|\delta_K) < \theta_l$, it determines that the two concepts are different. In the case when $\theta_l \leq P(X = \mathcal{L}|\delta_K) \leq \theta_h$, they system will generate an alert for human intervention.

G MRF Algorithm

The returned results from different data sources are first transformed into the target ontology via SM function. Resource Description Framework Schema (RDFS) semantics [54], for example, can be used in the target ontology. MRF then performs duplicate detection to identify and mitigate duplicate information retrieved from different data sources. It then utilizes the MapReduce for reasoning and inferencing. In addition, MRF uses a trust value indicates the accuracy of the mining results at different resources to finally yield a fusion result. MRF detailed steps are described below.

1 Duplicate Detection (DD)

The first step of the IFE is to detect duplicate of information, i.e., two retrieved objects refer to the same real-world entity. This problem is critical in scenarios where information fusion query returns a large number of sources; consequently, the number of duplicate objects can explode and render the query result practically useless to the user. The approaches which deploy *overlapping attributes* may fail in identifying a correct duplicate. For example, we have first tuple $(9, John)$, where the first entry indicates the age and the second entry indicates the name, and another tuple $(John, Camry\_2003)$, from other data source, with the first entry represents the name and the second one represents the car possessed. In this case, the two tuples would not be matched because a match would result in a "John" who is 9 year old and has a car (i.e., Camry 2003). This is unlikely, based on our knowledge. Our proposed DD algorithm is to exploit the correlation and contradiction among the attributes of the objects to identify duplicate. In the above example, the contradiction from the attributes can be used to resolve the issues in the existing approaches.

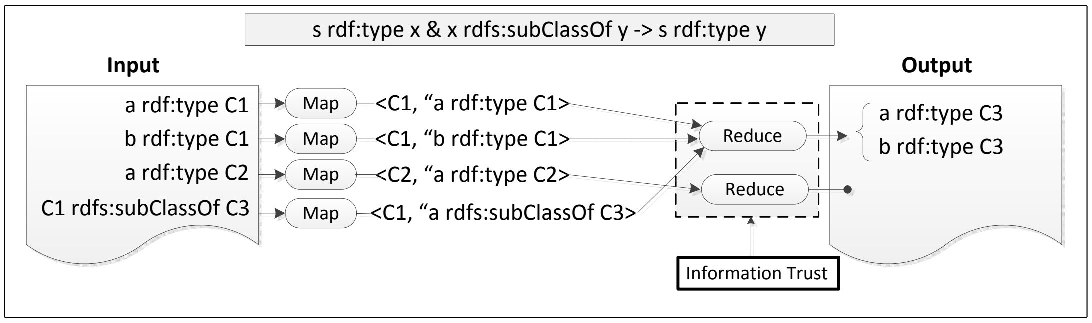


Figure 26. An Example of Sub-class Reasoning Using MapReduce.

2 MapReduce-based Reasoning

We now describe how to perform reasoning using the MapReduce [55] approach. Basically, the algorithm is performed in two steps:

- *Map*:

The *Map* processes each input triple and outputs a key/value pair. Note that to perform the sub-class join, triples with $rdf : type$ should be grouped on their object, whereas triples with $rdfs : subClassOf$ should be grouped on their subject.

- *Reduce*:
  The *Reduce* collects all tuples with the same key to compute the result. We note that this process will take into account the trustworthiness of the information sources for fusion and inferencing.

An Example

To illustrate the idea of using MapReduce method for reasoning, we show an example in Figure 26. Consider a rule of RDFS which derives $rdf : type$ based on the sub-class hierarchy. The Map procedure converts input triples into key/value pairs such as $< C1, ``a \quad rdf : type \quad C1'' >$, where the first entry is the key, and the second entry is its corresponding value. The Reduce procedure groups all intermediate results which have the same key for reasoning and inferencing. For example, based on $< C1, ``a \quad rdf : type \quad C1'' >$ and $< C1, ``a \quad rdfs : subClassOf \quad C3'' >$, the Reduce procedure then can reason that $``a \quad rdf : type \quad C3''$.
The other rules of the RDFS can also be implemented in the same manner.

3 Information Trust

When integrating information from different sources which may come from different origins; thus, their trustworthiness (i.e., accuracy) should be taken into account. For example, personal information of a person from the police record is usually more accurate than that on social network profile. We first start with the definition of trust in the context of computer networks.

Definition G.1 *(Trust). The trust is mathematically defined by function $\omega_E : E \times E \rightarrow \Sigma^E$, where E is a set of entities, Σ^E is the set of all possible trust values, and ω_E models trustworthiness T_E of all entities in E.*

In our system, we consider two levels of trust:

- Data Source Trust:

  This represents the trustworthiness of the data sources. Let S be the set of all data sources, their trust is defined as $T_S : S \rightarrow [0, 1]$, where higher values of T_S represent more trustworthy source. The true value of T_S for each data source has been derived from social networks in Eq (8).

- Semantic Relation Trust:

  This describes trust in attributes of an entity. This trust value will depend on the data source and is defined as $T_{A_s} : A_s \rightarrow [0, 1]$, where A_s is the set of attributes for data source $s \in S$. Similarly, higher values of A_s represent more trustworthy attribute. This depends based on the profession, location and network of people.

By combining the trust on these levels, the system computes the global trust value for information integration as $T = T_S \oplus T_{A_s}$, where $\oplus$ denotes the concatenation function, which can be a simple multiplication or some fuzzy logic operation). The global trust value will be used for inferencing during the Reduce procedure.

4 Using Trust to stop spammers

In the following cases we show how we can use the defined trust to stop the spammers and increase the reliability of the information fusion. The guy in the black dress with a hat is the spammer. The green files are good files and the red files are the spam files.

Case 1 - Spammer outside of your network:

Assume that the spammer is not directly or indirectly (through friends and family) related to the user as shown in Figure 27. The data source trust score for this user is low as he is not in our social circle and as a result the spam files from this spammer will have a very low probability to be included in the fusion results.

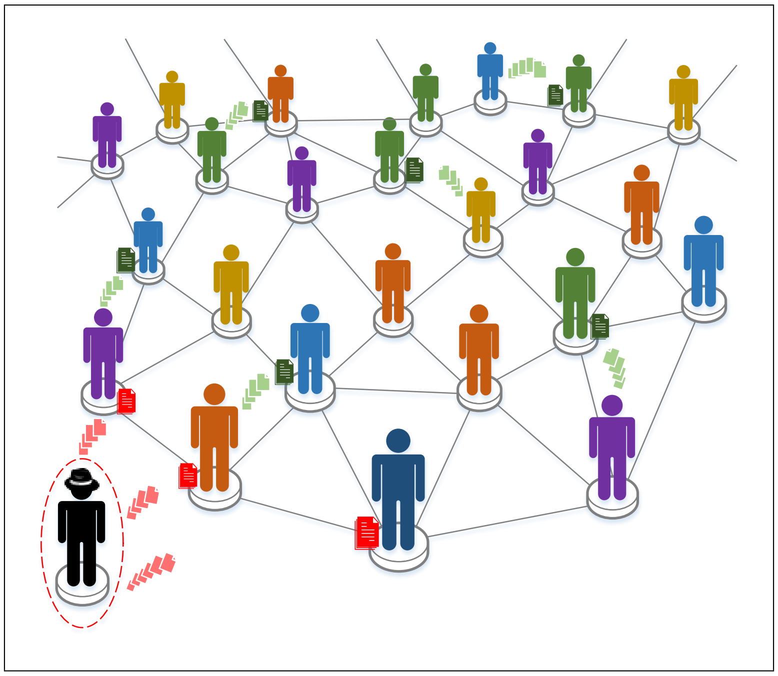


Figure 27. Spammer spamming from outside your social network

Case 2 - Spammer inside your network:

Assume that the spammer is directly or indirectly connected to our network as shown in Figure 28 Initially the spammer has a high trust and his spam might be included in the fusion results if he stuffs relevant keywords in his spam files. However over a period of time the data source trust keeps decreasing and after a threshold, the data source will not be used anymore.

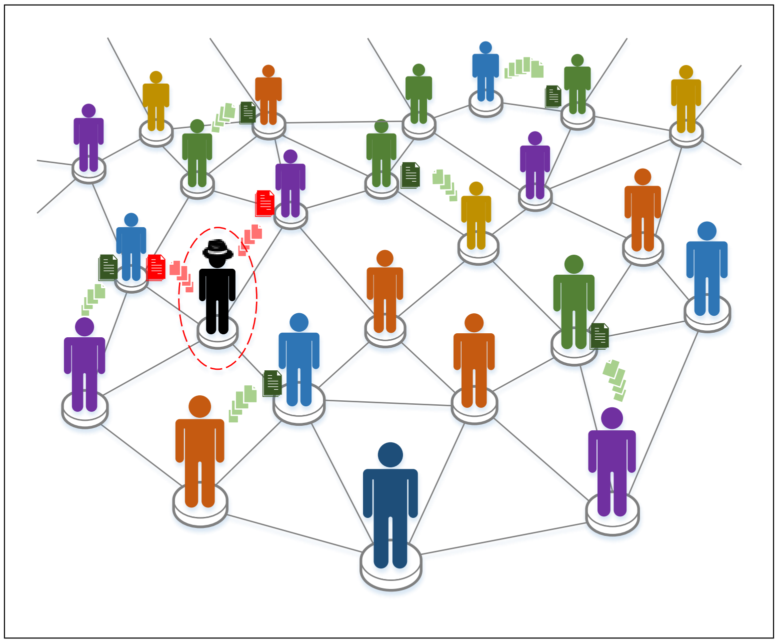


Figure 28. Spammer spamming from inside your social network

Case 3 - Intruder compromising a good node:

Assume that a spammer has compromised a user's account and is spamming from his account without the userś knowledge as shown in Figure 29. Since the user has a good reputation, initially the spammer will be able to send spam files. But with suddent spikes of uploading files, we reduce the data source trust and also based on the content. The system will be able to exclude this user's files from the fusion.

Case 4 - Unintentional Spamming by regular users:

Assume that regular users have some spam files on their machine and are uploading unintentionally as shown in Figure 30. This type of unintentional spamming is the hardest to detect as there won't be any surges in file uploading and the trust of the data source is high too. Here we have to assign weights to the document based on the content and user feedback and try to eliminate these type of spaam files.

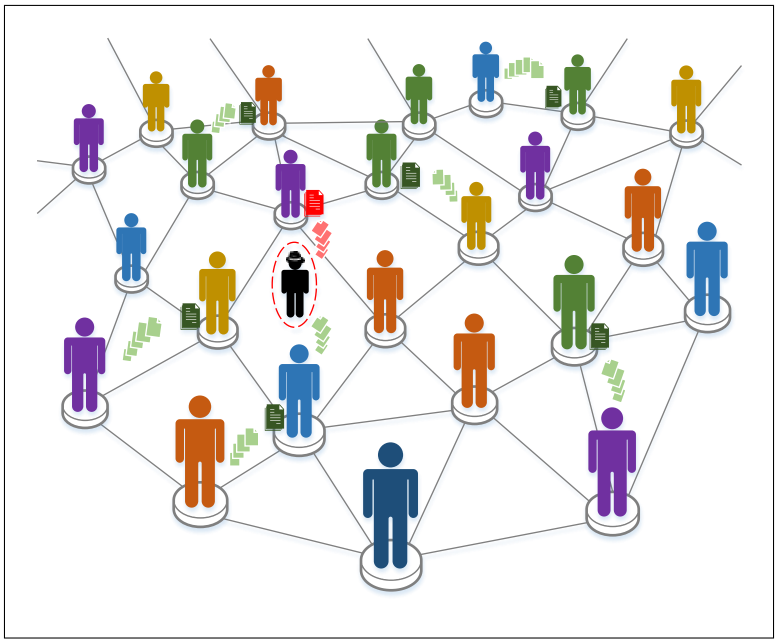


Figure 29. Spammer spamming after compromising a good node

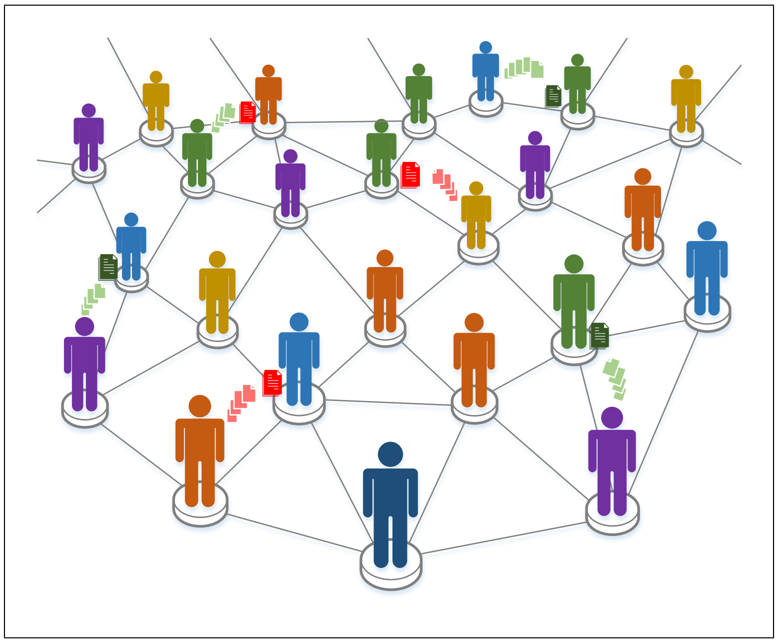


Figure 30. Unintentional spamming from users

5 Why Iterative Processing?

Typically, users do not know about the detailed collective information and the retrieval environment. At first, it's difficult for the users to formulate a query that well satisfies the information need, and then they again and again refine the queries until the retrieved information satisfies their need. This process requires intensive efforts from users to come up with appropriate modifications (e.g., changing keywords, providing additional keywords, etc.) In addition, it is more challenging in the environment of heterogeneous data sources because an in-appropriate query modification might retrieve more information from a data source, but it can also filter out important relevant information from other data sources. Therefore, we propose an iterative processing algorithm where the integrated information is fed back to distributed data miners so that they can find additional relevant information. In this way, the information obtained from one source can be used as a new clue for other sources retrieve additional information for the next iteration. The process is carried out until information returned is satisfied or number of iterations is reached as long as the accuracy does not fall below the specified threshold.

H Fusion Precision and Other Metrics

In our system, the quality of the fusion engine is quantified via the quality of the building components of the system.

Fusion Precision: We use *precision* and *recall* to quantitatively evaluate the quality of our data mining function. *F-measure* will also be used to combine the two metrics in a single figure. Let tp, fp, and fn, respectively, be the count of *true positives*, *false positives*, and *false negatives* of the results. We define the following metrics.

- *Precision*: is the fraction of the retrieved documents that is relevant $P = \frac{tp}{tp+fp}$.
- *Recall*: is the fraction of relevant documents that is retrieved $R = \frac{tp}{tp+fn}$.

- *F-measure*: is the weighted harmonic mean of P and R to combine both of them into a single score $F = \frac{2\times P\times R}{P+R}$. The larger value of F indicates a better IE system.

Additionally, the non-interpolated average precision defined by NIST will also be used for evaluating the retrieval effectiveness of the SDML function. We denote T be the number of true relevant documents in a set of size S; L the ranked list of documents returned; R_j the number of relevant documents in the top j documents. In addition, we define the indicator function $I_j = 1$ if the jth document is relevant and 0, otherwise. The non-interpolated average precision (AP) is then computed as:

$$AP = \frac{1}{T}\sum_{j=1}^{S}\frac{R_j}{j}\times I_j. \qquad (25)$$

CHAPTER V

PERFORMANCE ANALYSIS, IMPLEMENTATION AND EVALUATION

In this section, we will evaluate the performance of the proposed SDSF via both analysis and simulations. Particularly, we first theoretically analyze the reliability of the proposed SDSF and evaluate the system performances via real network implementation. In the second part, we show the performance gain of our proposed SDSF over the P2P-based systems.

A Implementation

The architecture of the SDSF is shown in Figure 31. SDSF system uses n layered architecture. We have a presentation layer, web service, DDS Controller and Data Layer. By using a layered architecture it is easier to manage the code as the changes in one layer will not affect the other layers. By using a web service, any device (using any platform) can talk to our web service. This way we do not have to re-write the code for different platforms. The user can access the SDSF system using a browser or through the SDSF application. The presentation Layer has the user interface. Users accessing through browser interact with the presentation layer. The presentation layer can interact only with the web service. Users accessing through the SDSF application directly interact with the web service. The DDS Controller has the business logic, the erasure coding and the Social Storage Provider functionality. Data Layer controls the access to database and storage nodes. Figure 31 shows how the Data Layer interacts with the storage nodes. The user uploads a file to the web service through the presentation layer. The web service sends the request to the Controller. Here the Controller authenticates the user and sends the status back to the web service. If the user is authenticated, the user uploads the file to the web service. The file is sent to the DDS Controller. DDS Controller performs the erasure coding and gets the storage providers from the social network. The data access layer then communicates with the storage nodes.

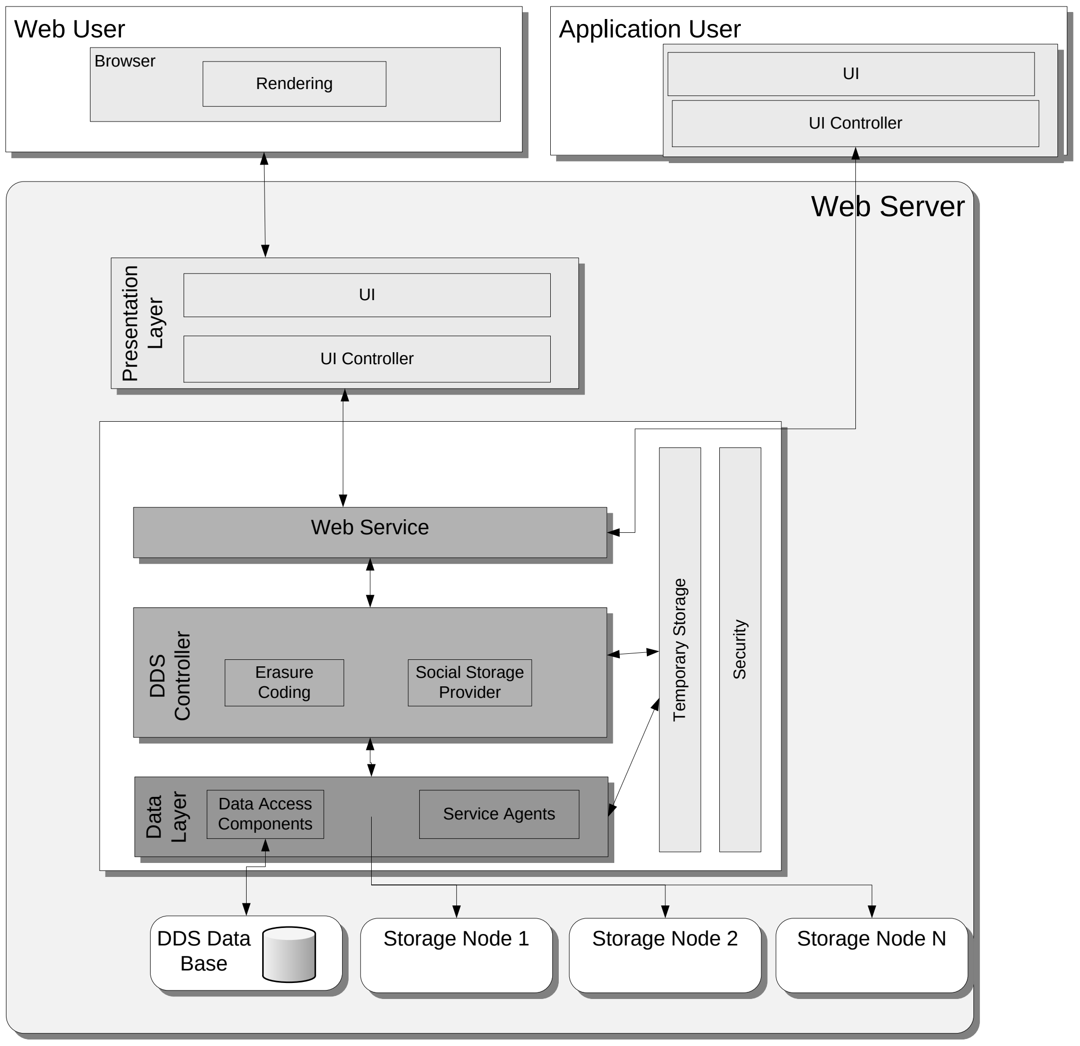


Figure 31. Architecture Overview

Figure 32 shows how the data is stored and retrieved from the storage nodes.

The service agents in the data access layer interact with the application on the storage nodes.

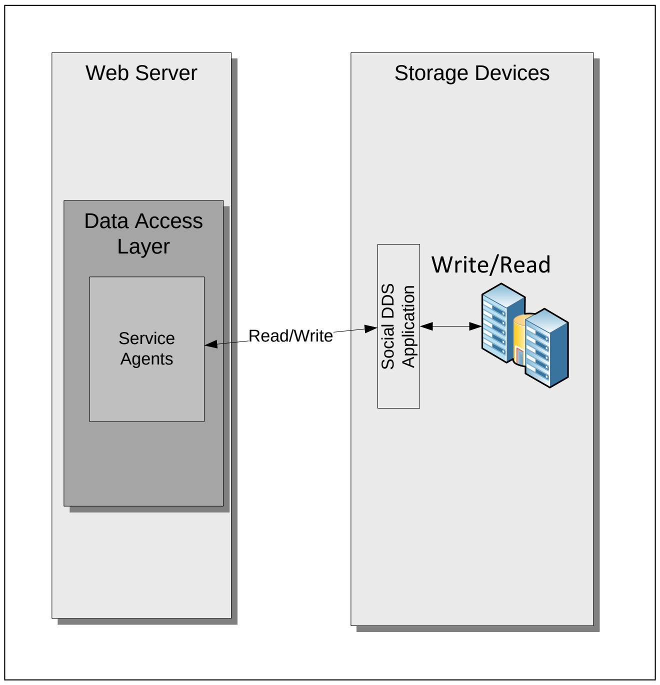


Figure 32. Accessing storage nodes

Figure 31 shows the architecture of SDSF mobile application and desktop application. The SDSF application directly communicates with the web service. The advantage of using a web service and layered architecture is that any changes in the business logic layer will not affect the clients.

B Storage Performance Evaluation

Basic setup

To measure the encoding and decoding performance, we used the JErasure codes written in C++ by James S. Plank [56]. We wrote a C# wrapper for the Jerasure codes using Visual Studio 2012 and ran them on a PC which is running Windows 7 (64 bits) with an Intel Pentium G 630 2.7 Ghz processor and 6 GB Ram. The social storage system was built by us using C#, Asp.Net, SQL Server 2012 which is hosted on IIS 7 Web Farm [57] and windows server 2012. We have used a web farm for scalability, and for testing we have used 4 servers in our web farm. We have used a database cluster for reliability of the database. The Social Storage system was built using the n layered architecture. The client from his browser calls the web farm which directs it to one of our servers which in turn does the file uploading, downloading, erasure encoding, decoding, and returns the status to the clients browser. We use the Monte Carlo method to evaluate the system performance where each experiment is performed many times and the average is computed.

1 Encoding and decoding performance

In this subsection, we show the system performance in terms of encoding and decoding runtimes via real network experiments with different settings. In our experiments, the runtime is the delay of encoding or decoding operations. Figure 33 shows data encoding and decoding runtimes of the $(8, 4)$ coding schemes for data files of various sizes. We see that the runtime of encoding files is much higher than the runtime of decoding files. We also see that the runtimes of encoding and decoding increases almost linearly with the file size.

Figure 34 shows data encoding and decoding runtimes of the $(12, 1, k)$ coding schemes on a 125 MB data file. As expected the data encoding runtimes is higher than the data decoding runtimes. Also as shown, the data encoding and decoding

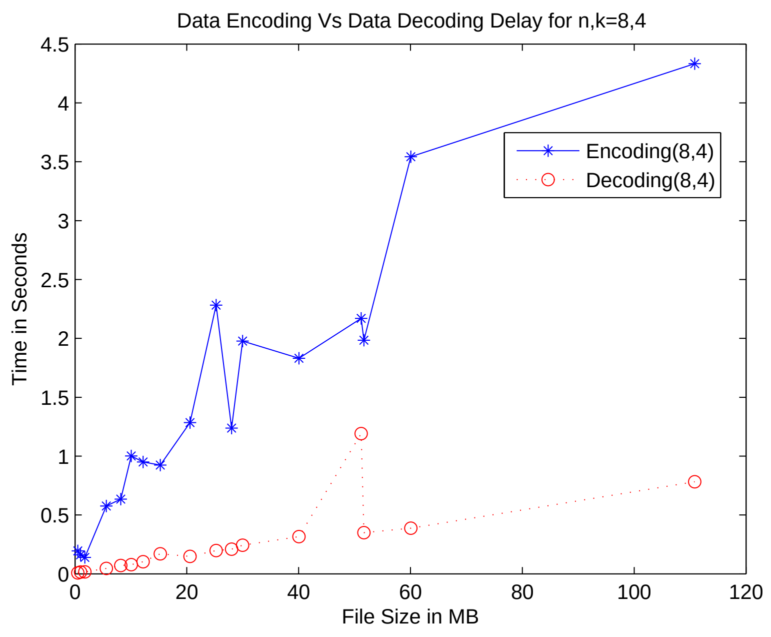


Figure 33. Encoding and Decoding runtime of $(8, 4)$ code vs. the file size.

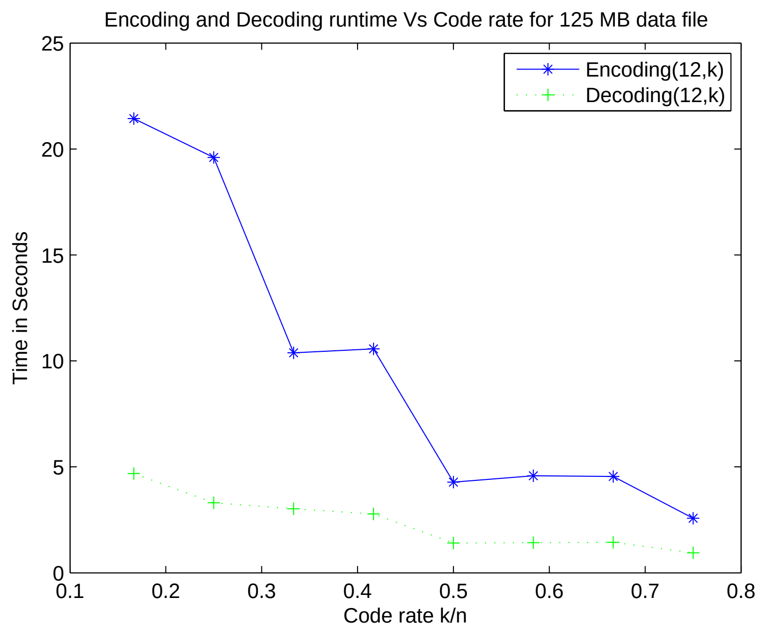


Figure 34. Encoding and Decoding runtime of $(12, k)$ codes vs. the code rate for 125MB data file.

runtimes decreases with the increase of the code rate. The reason is that when the code rate increases, the number of matrix multiplications are reduced; thus, it requires less computational load for data encoding and decoding.

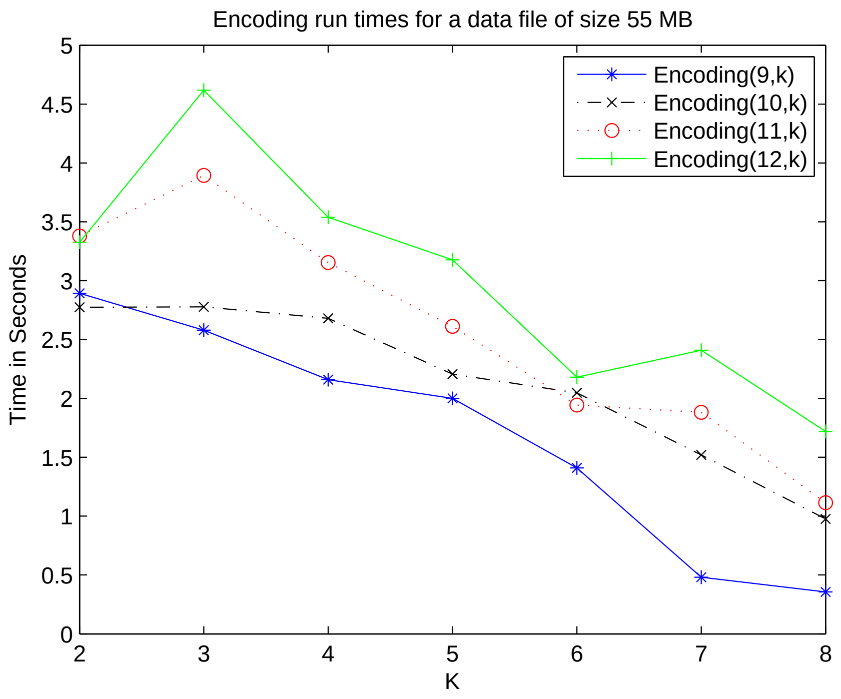


Figure 35. Encoding runtime of different (n, k) codes vs. k for 55 MB data file.

Figure 35 compares the encoding runtimes of (n, k) for a data file of size 55MB. We compare for n=9, 10, 11, 12 and k from 2, 3..8 keeping the file size constant. We observe that as n increases, encoding runtime increases and as k increases, encoding runtime decreases. Particularly, we observe that the encoding and decoding runtime increases with n. The intuition is that when n is small, the program performs computation on smaller size matrix, resulting in smaller runtime. On the other hand, when n increases, it requires more time to operate on a larger matrix. However, we note that when n is smaller, the failure probability of the system increases as the level of resiliency decreases. Thus, there is a trade-off which depends on the system characteristics such as number of storage nodes and transmission bandwidth.

Figure 36 compares the decoding runtimes of (n, k) for a data file of size 55MB.

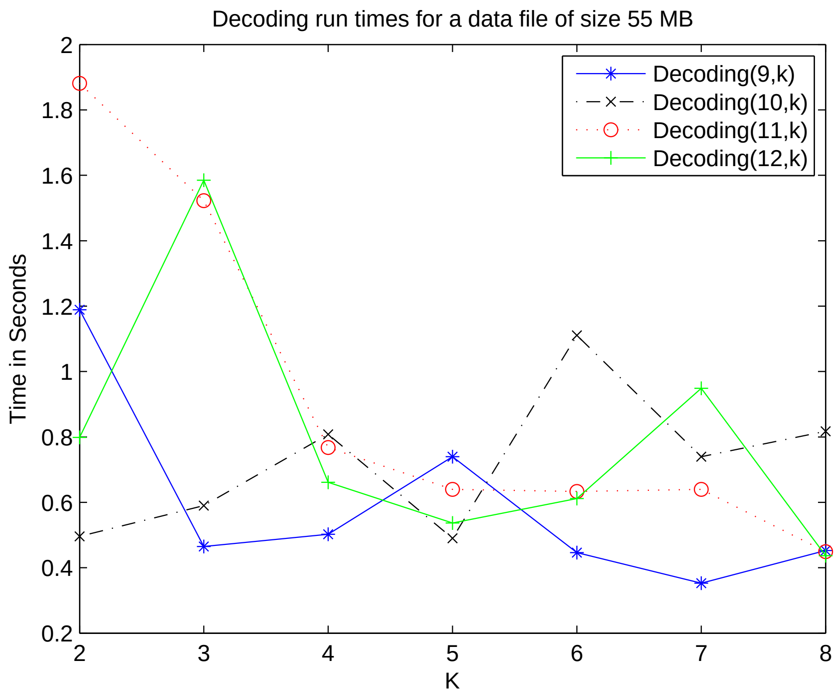


Figure 36. Decoding runtime of different (n, k) codes vs. k for 55 MB data file.

We compare for n=9, 10, 11, 12 and k from 2, 3..8 keeping the file size constant. We observe that the decoding runtimes do not follow any clear trend with n and k. Overall it appears that decoding runtime decreases with increase in k. There is not much we can deduce with n. We observed that since the runtime was very less the patterns were not clear. So we increased the file size to 125 mb and repeated the same test.

Figure 37 compares the encoding runtimes of (n, k) for a data file of size 125MB. We compare for n=9, 10, 11, 12 and k from 2, 3..8 keeping the file size constant. We again observe that as n increases, encoding runtime increases and as k increases, encoding runtime decreases. The pattern is very clear for encoding.

Figure 38 compares the decoding runtimes of (n, k) for a data file of size 125MB. We compare for n=9, 10, 11, 12 and k from 2, 3..8 keeping the file size constant. We observe that the decoding runtimes have a better trend than the smaller file size 55 MB. We can clearly see that as K increases, the decoding runtime decreases. However for N, it is still not very clear, although it looks like as N increases, decoding runtime

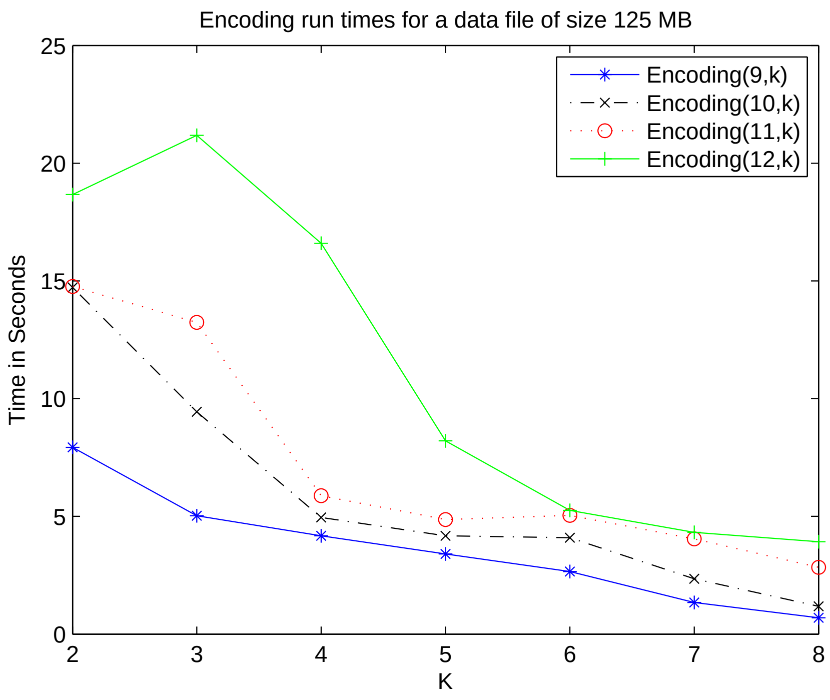


Figure 37. Encoding runtime of different (n, k) codes vs. k for125 MB data file.

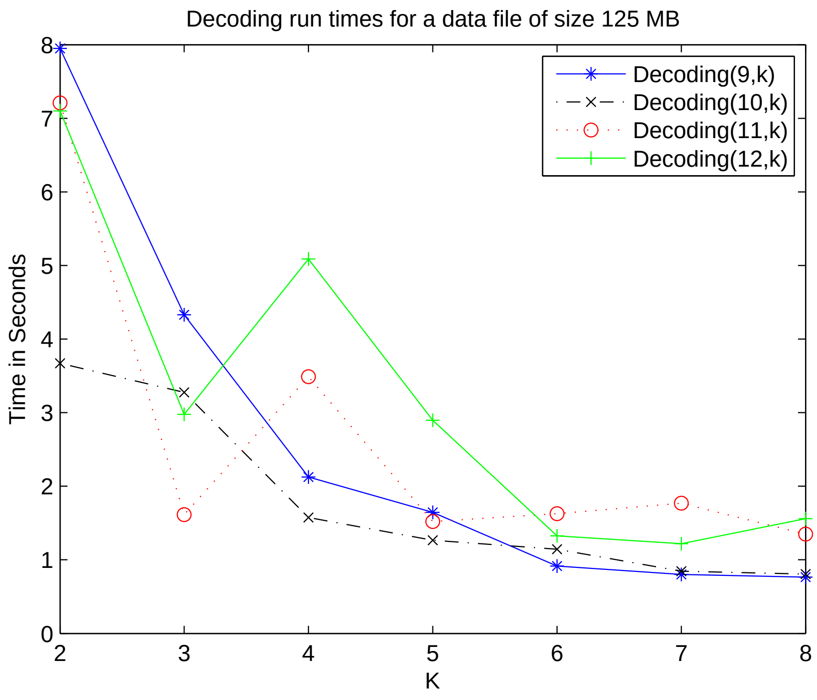


Figure 38. Decoding runtime of different (n, k) codes vs. k for 125 MB data file.

also increases.

From the above figures as expected, the $(8, 4)$ coding scheme outperforms all the others. This is because with the $(8, 4)$, the system operates on a smaller matrix size, resulting in a lower computation runtime. Additionally, we observe that the runtime of our implemented algorithm increases linearly with the data size; thus, is is feasible to encode larger data files with slightly increase of the runtime.

2 End-to-end delay

We evaluate the end-to-end delay of the proposed SDSF system in a real network implementation. The SDSF system was built using the n layered architecture. The client uses his browser to call the web service on our server which in turn does the file uploading, downloading, erasure encoding, decoding, and returns the status to the client's browser. Particularly, several files with different data sizes are used to writing and downloading from a users over the Internet. The user uses a laptop

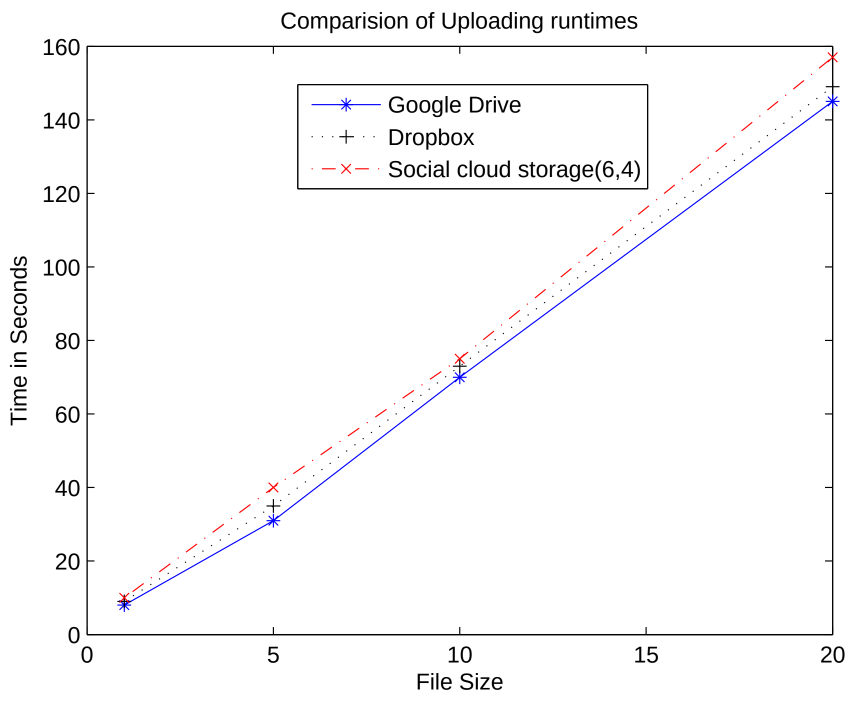


Figure 39. End-to-end delay of data writing/uploading in a real network.

connects to a gateway server, which is connected with multiple storage nodes. In this experiment, all data writing and reading are performed via the gateway server before sending to the storage nodes and user. We compare our system with the leading storage systems Dropbox and Google Drive.

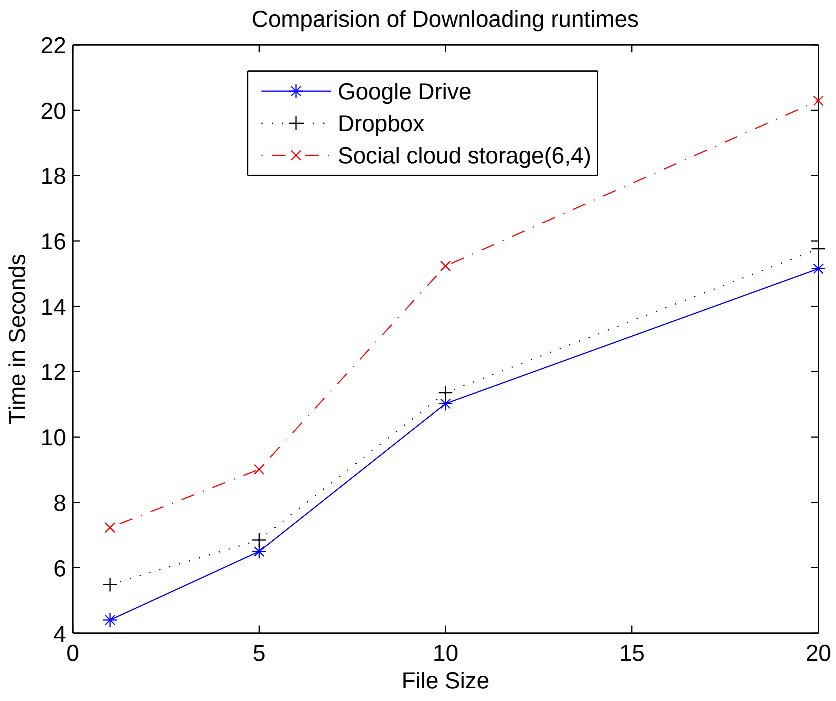


Figure 40. End-to-end delay of data reading/downloading in a real network.

The average end-to-end uploading delays are depicted in Figure 39 and the average end-to-end downloading delays are depicted in Figure 40. As shown, data writing delay is greater than that of the reading delay. The reason is due to the fact that the uploading speed of the Wi-Fi connection is less than that of the downloading connection. The machine we used for testing had an upload speed of 0.8 MBps and a download speed of 10MBps. In addition, we observe that the end-to-end delays also increase linearly with the data size.

C Fusion Performance Evaluation

1 Word Count Performance

We perform the word count for keyword indexer and for N-gram query correction. We have used Amazon Elastic MapReduce (Amazon EMR) [58] which is a web service used to process vast amounts of data. Amazon EMR has a resizable cluster of virtual machines and uses Hadooop map reduce. We used the following python word count program [59].

```
#!/usr/bin/python
import sys
import re

def main(argv):
  pattern = re.compile‘‘[a–zA–Z][a–zA–Z0−9]’’)
  for line in sys.stdin:
    for word in pattern.findall(line):
      print ‘‘LongValueSum:’’ + word.lower() + ‘‘\t’’ + ‘‘1’’
if __name__ == ‘‘__main__’’:
    main(sys.argv)
```

Figure 41 shows the execution time of a file with size 27 MB. We have tested the word count program with one VM, two VMs, and four VMs. MapReduce helps in breaking up a file of any size into very small pieces and process them individually in parallel. We can increase the machines available in the map reduce and reduce the execution time drastically. We can see from the figure that as we add more VMs, the runtime reduces drastically. Amazon EMR is highly scalable and we can add more VMs if we want to reduce the execution time even further.

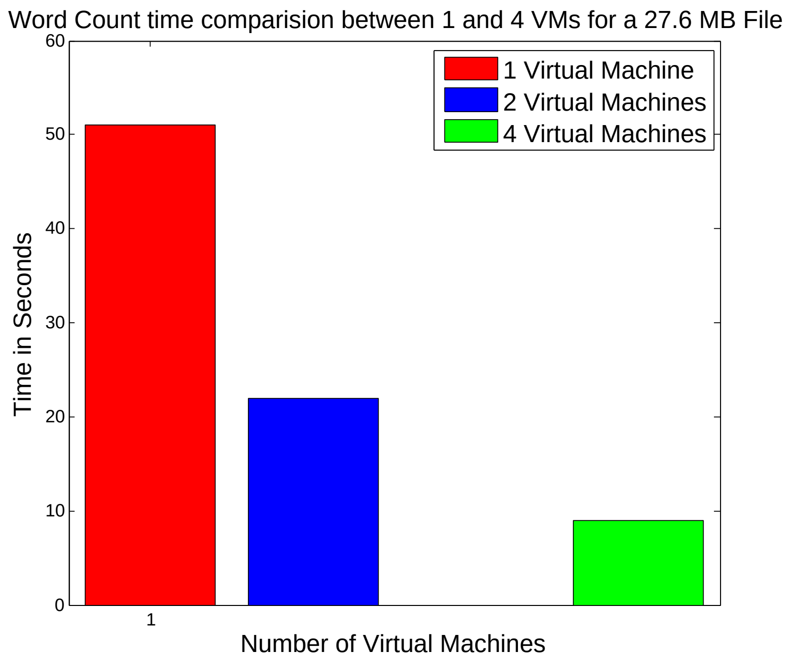


Figure 41. Word count runtime based on the number of VMś

2 Spam Filtering Comparison with and without Trust

We performed an experiment where we randomly mark some nodes in a network as spam, and these nodes keep producing spam with lot of keywords and useless or false material. Figure 42 shows the comparison of the results of Information Fusion with and without trust component. In our experiment we assume that even friends and acquaintances upload some spam and false data both intentionally and unintentionally.

3 Query Correction Comparison

We performed a query comparison for 100 queries with Bing, Google and SDSF system. Our SDSF system uses the Bing N-gram service and uses context knowledge on top of it to further understand the query. Table 3 shows a sample of our queries. The mistakes are highlighted with an underline. The user query 'bards' is correctly identified as 'birds' with the use of context knowledge. However context knowledge

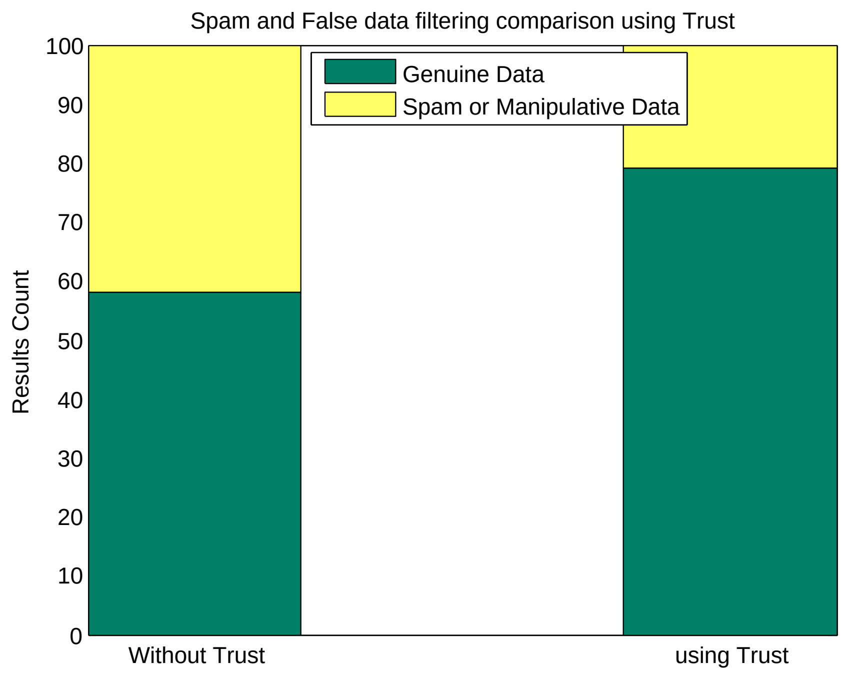


Figure 42. Spam Filtering comparison with and without Trust

can sometimes be a disadvantage too. We wrongly identified the user query 'buy flowr' as 'buy flour' because the user has a background in cooking and restaurants. The user query 'I would like to know the full rices of loiusville' has been wrongly identified by all three systems.

Table 4 shows the comparison of 100 sample queries. We can see that the number of correctly identified queries are higher in SDSF system than Google and Bing. However the queries which did not need correction, but were still corrected were the highest in SDSF. This was because our system uses the users' context if the query is not well formulated. The context can sometimes mislead our query correction.

D System Failure Comparison

We perform an experiment where we assume that nodes with lower social connections has higher failure probability and nodes with higher social connections has lower failure probability. In P2P network simulation, we assign random storage

TABLE 3. Sample query corrections

User Query	User Intention	Google	Bing	SDSF
Vehicle thefs	Vehicle thefts	Vehicle thefts	Vehicle theft	Vehicle thefts
bards	birds	bards	bards	birds
buy flowr	buy flower	buy flower	buy flower	buy flour
swim earn how to	learn how to swim	swim learn how to	swim learn how to	swim learn how to
I would like to know the full rices of loiusville	I would like to know the fuel prices of louisville	I would like to know the full prices of louisville	I would like to know the full prices of louisville	I would like to know the full prices of louisville

TABLE 4. Comparison of query correction systems over 100 sample queries

	Correctly Identified	Incorrectly Identified	Not needed
Google	81	19	4
Bing	78	22	3
SDSF	85	15	11

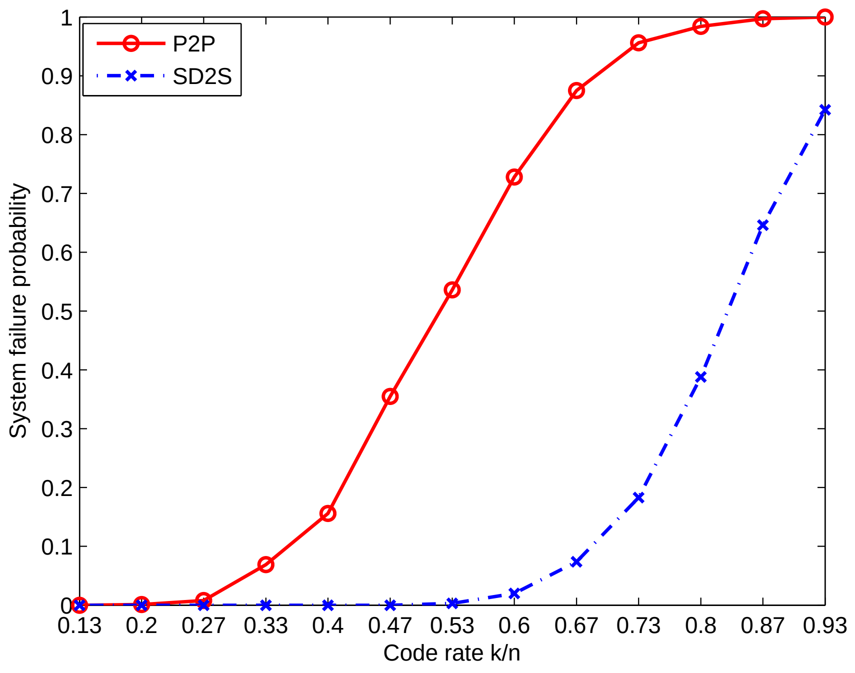


Figure 43. System failure probability vs. code rate.

nodes, while we assign storage nodes in SDSF nodes based on trust. Figure 43 shows the system failure probability of the P2P-based and SDSF systems for $(15, 1, k)$ code versus the code rate. In this simulation, we consider special case where the data source is a neighbor of the storage nodes. We focus on evaluating the impact of social ties on the system reliability. With that said, in this evaluation, we assume that each storage node has sufficient space for data storage. In addition, we assume that the source node has $N = 50$ social friends, the average social friends in Facebook [60], with the social ties uniformly distributed in [0,1]. Furthermore, we assume that storage node with smaller social tie value (i.e., the weaker relationship with the data source) has higher data failure probability. This assumption accounts for the "real life" relationships among the social users. The P2P system selects the storage node randomly while our proposed SDSF utilizes the social tie computations to select the storage nodes. As shown in Figure 43, the proposed SDSF system significantly outperforms the P2P system, especially in the regime of medium code rate k/n. This is because in this region, the probability of failing more then $n-k$ storage nodes in the SDSF is much lower than that of the P2P system due to the higher storage reliability, resulting from higher trusted nodes. On the other hand, in the regimes of low code rates, two systems obtain similar performance, i.e., low failure probability. The intuition is that in this region, the systems have high data resiliency $n-k$. On the contrary, in the high region of code rate, the failure probability of the two systems increases substantially due to the low resiliency factor. In this case, even exploiting the high trusted storage nodes, the chance data is unrecoverable in the SDSF is still high.

CHAPTER VI

SDSF SCALABILITY

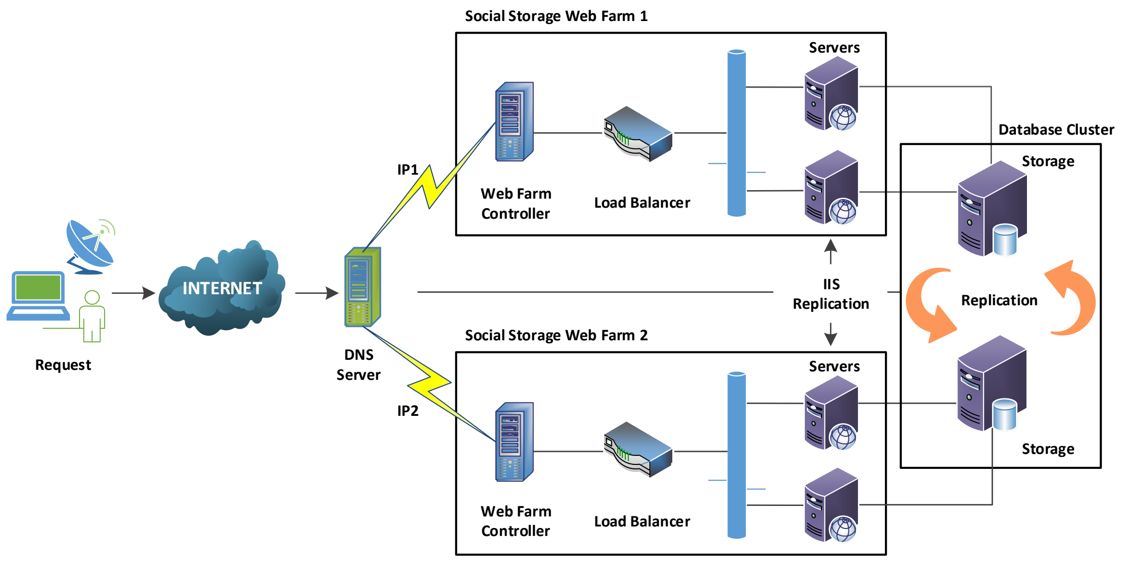


Figure 44. System scalability architecture

We wanted to build a scalable system so that it can grow and serve any number of users without any upper bound. For this purpose we implemented the use of IIS web farms and DNS load balancing. Figure 44 shows the system architecture with two web farms. Each web farm has one controller and X number of servers. The system can have any number of web farms. For our host name we assign multiple IP's (the IP's of each web farm controller) in the Domain Name Server (DNS). In this example we assign two IP's as we have two controllers. When the user first makes a request to our social storage system, the DNS server will randomly select an IP(IP of the web farm controller). The request comes to the web farm controller which uses the load balancer and assigns it to one of the available servers. The request is served by these servers. The load balancer can use one of the available algorithms [61] like:

1. Weighted Round Robin: Distributes requests to the servers based on the assigned weights. By default all weights are equal. So all servers get equal number of requests.

2. Weighted Total Traffic: Distributes requests to the servers based on the request and response size. Which ever server has the least amount of data, request is forwarded to it.

3. Least current request: Distributes requests to the servers based on the server with the lowest number of requests.

4. Least response time: Distributes requests to the servers based on the fastest response time.

5. Server variable hash: Distributes requests to the servers based on the server hash value.

6. Query string hash: Distributes requests to the servers based on the hash value of the query string.

7. Request hash: Distributes requests to the servers based on the configured hash variable of the server.

We tested various load balancing algorithms and found that weighted total traffic worked best for us as our servers were all of the same configuration and the execution time depends on the data size. So if data size is evenly distributed the load is also evenly distributed.

A System Failures

The system is built such that there is no single point of failure. Any server can fail at any time including the web farm controller, processing server or the database server. The system does not rely on them always working. Let us see the possible machine failures:

- Processing Server Failure
  Assume that a server within a webfarm fails. The web farm keeps pinging the processing server every t seconds, where t is the pre defined time interval we want the web farm controller to ping the processing server. If there is no response from the processing server, the web farm controller will stop all requests to the failed processing server. As the system grows we can have as many processing servers under a web farm controller as we want.

- Web Farm Controller Failure
  Assume that a web farm controller has failed. When the user types in our host name, the DNS server would translate the host name to one of the IP's of the web farm controller. When the requests to that IP keep failing, DNS server would stop translating the host name to the failed IP of the web farm controller. As the system grows we can have as many web farms in different geographic locations as we want.

- Database Server Failure
  We are currently using Always on availability group [62] of Sql Server which uses five database servers with up to two synchronous synchronizations and three more for asynchronous synchronizations. As we have two databases with simultaneous synchronizations, read operations can be performed from both the servers. It can handle up to four database server failures.

CHAPTER VII

CONCLUSION AND FUTURE WORK

A Conclusion

1 Data Storage

In this proposal, we present a system design for a distributed data storage system based on the social networks. In particular, our proposed system implements erasure-based coding and decoding where the original data file is encrypted before dividing into multiple small pieces which are then encoded into many coded fragments. The coded fragments are reliably distributed to the storage nodes. Different from the P2P systems, our proposed SDSF system chooses the storage nodes based on the "social trust" which is computed from the interactions in the social networks. To tackle the problem of finding the potential storage nodes in a large size social network, we proposed an efficient algorithm to compute the social ties among the social nodes. Furthermore, our proposed system also allows the data to be regenerated when a storage node fails or leaves the network. This process ensures that at any time the data in the network is sufficient for reconstructing the original file. Our theoretical analysis and simulation results show that the data stored in our proposed SDSF system is reliable with a failure probability approaching zero when more redundant data is added. Particularly, our simulation results show that data reliability of our proposed SDSF system is higher compared to the P2P systems.

SDSF system does not necessarily have to be used with social networks. There are applications where we do not want our data to be stored in systems outside of our network for example medical data. SDSF system can be used by an enterprise to construct and manage the internal data network. There are other areas where our SDSF system could be used with minor changes:

Storage Service Provider

If a business wants to start a storage service like Dropbox or Google, they can use our SDSF software to manage data on their data servers. The SDSF system selects nodes from their data servers and stores the user data on these servers.

Large Data(e.g., GIS Imagery)

We have vast amounts of raw data being produced from the digital images of earth from satellites. Given the vast amounts of data involved we cannot store the information in one place. We can use SDSF to securely store large data while saving lots of storage space when compared to storing using file redundancy.

Personal Data Management

Personal users who have multiple smart devices (For example laptop, touchpad and smart phone) can create a cloud network to manage and utilize the space efficiently while enjoying all the benefits of cloud storage. Some users utilize lot of storage space in some devices while other storage devices are underutilized. These kind of users will benefit a lot as SDSF can distribute data based on device capacity. For example mobile devices have limited space. With SDSF System we can store major portions of the file on other devices owned by the user.

2 Information Fusion

We have conducted the set of experiments as a proof of illustration of success of Information Fusion. We create an Index server that maintains the keyword, their count and the location. Once this activity if performed, it need not be repeated unless a new node/document is included in the existing network. This will improve the performance as the indexing part need not be repeated. To illustrate that using Map reduce will enhance the Fusion performance, we demonstrated how use of Map Reduce reduces the run times of the process by running on multiple machines. The success of the Information fusion depends on the fact that the query of the user is accurate. We have demonstrated through various experiments how the "not so well formed" query can be corrected. We compared our results with Google and Bing and found that our system performs better than them due to the incorporation of context knowledge with the query (although in some cases, the context leads to an error). We also demonstrated that use of trust scores improves the retrieval of genuine documents which will effectively lead to better Accuracy of Information Fusion due to selection of a better candidate set. Although, the Information fusion using our method requires more time, the precision and accuracy of the results will increase due to use of trust scores. This is the tradeoff required for increased accuracy. Generally the Information Fusion technique has been utilized for fusing data obtained from wireless sensor networks. However, it can also be utilized for fusing the traffic conditions and weather data, Stereo vision, Fire detection using temperature, humidity and vision sensors or flood forecasting. Some novel applications would be to fuse data on various social networks and customer profile to make recommendations.

B Future Work

1 Improving Potential Storage Selector

We will work on improving the algorithm for selecting potential storage providers. Currently we are using the Dijkstra's algorithm k times to find the K paths to the storage provider. We will look at heuristic ways to decrease the runtime complexity of the algorithm. The heuristic algorithm might not give the best solution, but it will give the near optimal solution. We will also re-evaluate other parameters like bandwidth, availability, space, trust, etc.

2 Minimizing Fault Recovery Overhead

Currently we are using Minimal Storage Nodes for fault recovery. Here we recover the data from any k surviving nodes and write the data to a new node. Consider a File with size M bytes. To recover a storage node of size M/K we need to transmit M bytes. We will implement the minimal communication recovery along with the traditional minimal storage recovery so the bandwidth consumed will be less. This method requires additional computation but as fault recovery is a background process the user will not be affected much.

3 Expanding search functionality for images and other file formats

Currently we have implemented only the text based searching and we haven't implemented searching for other file formats like images. Image searching is a difficult problem as the image which is in parts does not give any information unless the image is combined. If we have to know the information from the split pieces we need to have the knowledge of all file formats which is not feasible. So we can instead extract all possible information from the file before splitting and saving the split files.

REFERENCES

[1] (2013, Jan.) Harvard business review. [Online]. Available: http://hbr.org/2012/10/big-data-the-management-revolution/ar/1 1

[2] M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: An efficient scheme for tolerating double disk failures in raid architectures," *IEEE Transactions on Computers*, 1995. 1

[3] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An architecture for global-scale persistent storage," in *Proceedings of the 9th IEEE ASPLOS*, 2000. 2, 18, 19

[4] B. W. I. Clarke, O. Sandberg and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," 2001. 2, 18, 20

[5] F. Dabek, J. Li, E. Sit, J. Robertson, M. Kaashoek, and R. Morris, "Designing a dht for low latency and high throughput," in *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, 2004. 2, 18, 21

[6] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, "Total recall: System support for automated availability management," in *Proceedings of Symposium on Networked Systems Design and Implementation (NSDI)*, 2004. 2, 18, 22

[7] https://www.dropbox.com/. 2, 23

[8] https://drive.google.com/. 2, 23

[9] R. R. Brooks and S. S. Iyengar, *Multi-sensor fusion: fundamentals and applications with software.* Prentice-Hall, Inc., 1998. 5

[10] T. Bass, "Intrusion detection systems and multisensor data fusion," *Communications of the ACM*, vol. 43, no. 4, pp. 99–105, 2000. 5

[11] C. Siaterlis and B. Maglaris, "Towards multisensor data fusion for dos detection," in *Proceedings of the 2004 ACM symposium on Applied computing.* ACM, 2004, pp. 439–446. 5

[12] E. F. Nakamura, F. G. Nakamura, C. M. Figueiredo, and A. A. Loureiro, "Using information fusion to assist data dissemination in wireless sensor networks," *Telecommunication Systems*, vol. 30, no. 1-3, pp. 237–254, 2005. 5

[13] C.-L. Yang, S. Bagchi, and W. J. Chappell, "Location tracking with directional antennas in wireless sensor networks," in *Microwave Symposium Digest, 2005 IEEE MTT-S International.* IEEE, 2005, pp. 4–pp. 5

[14] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems.* ACM, 2003, pp. 14–27. 5

[15] T. Fei, D. Kraus, and A. Zoubir, "Contributions to automatic target recognition systems for underwater mine classification." 5

[16] F. E. White, "Data fusion lexicon," DTIC Document, Tech. Rep., 1991. 6

[17] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, 1997. 6

[18] B. V. Dasarathy, "Sensor fusion potential exploitation-innovative architectures and illustrative applications," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 24–38, 1997. 6

[19] E. F. Nakamura, A. A. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," *ACM Computing Surveys (CSUR)*, vol. 39, no. 3, p. 9, 2007. 6, 8

[20] R. B. Ross, R. Thakur *et al.*, "Pvfs: A parallel file system for linux clusters," in *In Proceedings of the 4th Annual Linux Showcase and Conference*, 2000, pp. 391–430. 13

[21] P. J. Braam *et al.*, "The lustre storage architecture," 2004. 13

[22] R. Bhagwan, D. Moore, S. Savage, and G. Voelker, "Replication strategies for highly available peer-to-peer storage," *Future directions in distributed computing*, pp. 153–158, 2003. 14

[23] P. Sobe, "Adaptations of block layout in distributed storage systems," in *Parallel, Distributed, and Network-Based Processing, 2006. PDP 2006. 14th Euromicro International Conference on.* IEEE, 2006, pp. 10–pp. 14

[24] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," in *ACM Computer Communication Review*, vol. 27, 1997. 16

[25] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," *Peer-to-Peer Systems*, pp. 328–337, 2002. 16

[26] R. Rodrigues and B. Liskov, "High availability in dhts: Erasure coding vs. replication," *Peer-to-Peer Systems IV*, pp. 226–239, 2005. 16

[27] J. Wang, W. Gong, and C. Xie, "A quantitative evaluation model for choosing efficient redundancy strategies over clouds," in *Networking, Architecture and Storage (NAS), 2012 IEEE 7th International Conference on*, june 2012, pp. 217 –226. 16

[28] D. Leong, A. Dimakis, and T. Ho, "Distributed storage allocations," *Information Theory, IEEE Transactions on*, vol. 58, no. 7, pp. 4733–4752, 2012. 16

[29] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008. 17

[30] (2014, Aug.) Xoring elephants: Novel erasure codes for big data. [Online]. Available: http://anrg.usc.edu/~maheswaran/Xorbas.pdf 18

[31] (2014, Aug.) Xorbas, a modified hadoop hdfs with new erasure codes. [Online]. Available: http://smahesh.com/HadoopUSC/ 18

[32] "Bit torrent file sharing protocol," https://bitconjurer.org/bittorrent/. 18

[33] http://aws.amazon.com/s3/. 23

[34] "Summary of the Amazon ec2 and Amazon rds service disruption in the US east region," https://aws.amazon.com/message/65648/. 24

[35] "Microsoft cloud data breach heralds things to come," https://www.techworld.com/au/article/37211/. 24

[36] http://www.facebook.com/. 25

[37] https://twitter.com/. 25

[38] L. Cutillo, R. Molva, and T. Strufe, "Safebook: A privacy-preserving online social network leveraging on real-life trust," *Communications Magazine, IEEE*, vol. 47, no. 12, pp. 94 –101, dec. 2009. 25

[39] K. Chard, S. Caton, O. Rana, and K. Bubendorfer, "Social cloud: Cloud computing in social networks," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, july 2010, pp. 99 –106. 26

[40] K. Chard, K. Bubendorfer, S. Caton, and O. Rana, "Social cloud computing: A vision for socially motivated resource sharing," *Services Computing, IEEE Transactions on*, vol. 5, no. 4, pp. 551 –563, quarter 2012. 26

[41] R. Hull, D. Jenkins, and A. McCutchen, "Semantic enrichment and fusion of multi-intelligence data," *White Paper*, 2006. 27

[42] G. L. McDowell, *Cracking the Coding Interview: 150 Programming Questions and Solutions.* CareerCup, 2013. 40

[43] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, “Network flows: theory, algorithms, and applications,” 1993. 53

[44] (2013, Jan.) Shamir’s secret sharing algorithm. [Online]. Available: http://en.wikipedia.org 57

[45] http://windows.microsoft.com/en-us/windows-8/search-file-explorer. 58

[46] https://www.google.com/. 58

[47] http://www.bing.com/. 58

[48] R. Antony and J. Karakowski, “Fusion of HUMINT and conventional multi-source data,” *MSS National Symposium on Sensor and Data Fusion*, 2007. 60

[49] https://class.coursera.org/nlp/lecture/. 63

[50] D. Jurafsky and J. Martin, *Speech and Language Processing: An introduction to speech recognition*, MIT, Computational Linguistics and Natural Language Processing Lecture, 2007. 67

[51] Y. Ko, “A study of term weighting schemes using class information for text classification,” *In proceedings of SIGIR’12, ACM*, 2012. 72

[52] http://uima.apache.org/. 73

[53] https://uima.apache.org/sandbox.html#regex.annotator. 73

[54] P. Hayes, *RDF Semantics*, W3C Recommendation, http://www.w3.org/TR/rdf-mt/, 2004. 81

[55] http://hadoop.apache.org/docs/stable/mapred_tutorial.html/. 82

[56] J. S. Plank, S. Simmerman, and C. D. Schuman, "Jerasure: A library in C/C++ facilitating erasure coding for storage applications - Version 1.2," University of Tennessee, Tech. Rep. CS-08-627, August 2008. 93

[57] http://www.iis.net/downloads/microsoft/web-farm-framework. 93

[58] http://aws.amazon.com/elasticmapreduce/. 100

[59] https://elasticmapreduce.s3.amazonaws.com/samples/wordcount/wordSplitter.py. 100

[60] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph," in *http://arxiv.org/pdf/1111.4503v1.pdf*, 2011. 104

[61] http://technet.microsoft.com/en-us/library/dd443524 107

[62] http://msdn.microsoft.com/en-us/library/ff877884.aspx. 108

CURRICULUM VITAE

Name:	Phani Chakravarthy Polina
Email:	p0poli01@louisville.edu
Address:	Department of Computer Engineering and Computer Science Duthie Center Lab 220 University of Louisville Louisville, KY 40292
Education:	M.S. Computer Science Western Kentucky University 2006
Experience:	RTDSS, LODI, UOFL Aug 2010-July 2014 Advanced Interactive System, Jan 2007 - July 2009
Skills:	Operating Systems: Windows 7, 8, Windows Server 2012, Linux Databases: SQL Server 2008, 2012, MS Access Development Tools: Visual Studio 2012, Eclipse, Matlab, SAS Languages: C#, C, C++, Java, SQL, PL/SQL, LINQ Web Technologies: ASP.NET, MVC, HTML, ADO.NET, REST, JQuery Web Servers: IIS GIS Tools: Desktop GIS (Arc info)

Papers: “Polina, Phani C; Tran, Tuan T; Xie, Bin; Kumar, Anup;”,
SOS: Social network-based distributed data storage,
“Local Computer Networks (LCN), 2013
IEEE 38th Conference on”, 687-690, 2013, IEEE

“D’Souza, Darryl; Polina, Phani C; Yampolskiy, Roman V; ”,
Avatar captcha: Telling computers and humans apart via
face classification, “Electro/Information Technology (EIT),
2012 IEEE International Conference on”, 1-6,2012, IEEE

“Liu, Yang; Kelley, Robert; Polina, Phani; Heragu, Sunderesh;
Kumar, Anup; ”,
iResTrac: an architecture for maintaining medical resource
status with mobile devices,
Proceedings of the First ACM MobiHoc Workshop on Pervasive
Wireless Healthcare, 10, 2011, ACM

“Pandit, Anala; Polina, Phani; Kumar, Anup; ”,
CLOPRO: A Framework for Context Cloaking Privacy Protection,
“Communication Systems and Network Technologies (CSNT),
2014 Fourth International Conference on”, 782-787, 2014, IEEE

“Pandit A. A., Polina P., Kumar A., Xie. B.”,
“CAPPA: Context Aware Privacy Protecting Advertising Extension
to CLOPRO Framework,
11th IEEE International Conference on Services Computing June 27
- July 2, 2014, Anchorage, Alaska, USA, Accepted