Privacy-aware Publishing of Decentralized Access-Controlled Content

THÈSE N^{O} 5646 (2013)

PRÉSENTÉE LE 12 FÉVRIER 2013

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE SYSTÈMES D'INFORMATION RÉPARTIS

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Rammohan NARENDULA

acceptée sur proposition du jury:

Prof. R. Guerraoui, président du jury
Prof. K. Aberer, directeur de thèse
Prof. S. Buchegger, rapporteur
Prof. J.-P. Hubaux, rapporteur
Dr O. Verscheure, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2013

Dedicated to

The lotus feet of my beloved Gurujee
Sri Sri Paramahansa Yoganandajee
for His teachings full of Divine Guidance
&

My Parents
for their unconditional love and care
&

The Swiss Alps
for everything I learnt from them

Acknowledgements

January 3, 2007. The day- I set out on a clueless mission of pursuing my doctorate dreams. I left behind, every bit of what defined me hitherto, to land in a world of beauty and tranquility too artificial and inconceivable for a dusty, noisy, disorderly managed life of Hyderabad, India. Chilling winter cold of Swiss, home sickness, and jerky cultural shock from a foreign land, made me a patient in the journey of becoming a doctor! *January 3, 2012.* The day- I am tagged "Dr."! A proud moment worth lifetime celebration! The (very!) long journey has been proved to be a unique learning experience both on professional and personal front.

Undoubtedly, the entire credit goes to my beloved Gurujee, Sri Sri Paramahansa Yogananda. I owe a lot to Him for the inspiration and energy I received from His teachings, that drove me a long way, getting rid of every obstacle. All that I have achieved during my stay is just because of the infinite faith I have on the Creator, thanks to my Geetha Autobiography of a Yogi, and whatever the unforgettable things I did, if any, due to my ignorance, belong to me and me alone.

When I sit back and think about the people influenced me and helped me complete this thesis, I am overwhelmed! I would not have seen this day, was it not Prof. Karl Aberer, my thesis adviser. I would like to express my immeasurable gratitude for hiring me as his Ph.D student. I am deeply indebted to him for all the guidance, patience, care, and encouragement during my research tenure at EPFL. He was extremely patient during troubling phases of my Ph.D and I find no words that would suffice to express my gratitude towards him. He is exceptionally smart and passionate about ideas, building systems. The way he raises intriguing questions after giving a talk always makes me puzzled. Rarely one sees a senior professor writing code to interact with a database! Undoubtedly, he will continue to be a role model for me.

Thanasis G. Papaioannou was instrumental in every aspect of my Ph.D work. We spent countless hours discussing the technical details of each work as well as issues ranging from politics to personal life. I was fortunate to have him around when needed. His amazing hard work, commitment, and passion in collaborative works inspire me all the time. I can never forget how he took ownership of some tasks and walked extra miles to finish them. I thank Zoltan Miklos very much for his patience and collaboration in the beginning of my Ph.D. He is extremely disciplined in managing his work which is truly

inspiring. I had a very good collaboration with Berker Agir and he always amazes me with his rapid turnaround times in bringing ideas to life. I had the luxury of working with really bright masters and bachelors students. Nikola Markovic stands top of all. He has mesmerizing implementation skills which were priceless in prototyping our ideas. I thank all the TEAM project members especially Walid Maleej, Hans, Dimitris, and Ivan for giving me a flavor of wonderful international collaboration experience. I thank my thesis jury committee members Prof. Jean-Pierre Hubaux, Dr. Olivier, Dr. Sonja, and Prof. Rachid for their valuable efforts and time in evaluating my thesis.

My stay in Switzerland would be the best days in many years to come, thanks to wonderful friends and colleagues. I enjoyed every moment with Surender, Saket, Nishanth, Tewari, and Srini during hundreds of looong tea and lunch sessions with wonderful discussions on myriad of issues- gossips on girls to realizing India 2.0. Those special philosophical chats with Nishanth and Tewari and high-tech enthu discussions with Srini added true colors to my Swiss stay. Surender and Saket made my stay in LSIR truly memorable with their amazing friendliness and non-stop fun. Not a single moment of my stay went without fun and more fun, thanks to their joyous company. I will miss this gang of 5 the most. There must be some eternal bonding to be with Ganga both in M.S and then during Ph.D. His amazing energy levels and passion in everything he undertakes would continue to inspire me. Thanks Ganga! Mr. cool Anurag, Dude Arnab, Shourya, Kailash, Somi, Shashi, Chary, Sid, Murali, Soni, Nitesh, Anudeep, Vijay, Raja, Krishnan, Mukul, Avinash, Prasad and all the lovely people Devika, Aamani, Nicee, Swapna, Czuee, Payal, Chaavi– the list is endless. I had a number of lovely experiences with every one, leaving them behind and moving on seems impossible. YUVA, the Indian association gave me an opportunity to meet and work with many incredible people. Subbu, Dada, ASK, Kalyan even though not in Swiss kept me focused on Ph.D.

My wonderful experiences in LSIR would never be complete without amazing lab mates: Mehdi, for all the hikes and sports we did together, Dude Michele, for all the wonderful chats and help, LSIR Fellow Zhixian, for being an officemate, Tri, for all the fun and energy, Alexandra, for all her cool and well-composed technical discussions, Hung, Tian, Dipanjan- for being wonderful colleagues. I thank Chantal again and again for all the administration related help during my stay.

Words cannot express my gratitude to my family members. The hard work of my parents kept me in alert, every minute in all the tests of time, right from my childhood.

-N. Rammohan, 13-1-13, 1 AM.

Abstract

The peer-to-peer paradigm is increasingly employed for organizing distributed data resources for various applications, e.g., content publishing and distribution, open storage grid, and online social networking. Further, private and semi-private access controlled content on the network has grown rapidly in recent years particularly user- generated content thanks to the explosion of the Web 2.0 technologies. However, the conventional P2P data management systems have failed to keep pace to support access controlled content with search efficiency for instance, similar to that of the structured P2P systems. Hence, their adoption is limited by the lack of possibility to control the access on the resources shared in the system. Moreover, in open environments with untrusted peers, even when proper access control mechanisms restrict the access to the resources, privacy issues may arise depending on the application, for example, an application to build an index of access controlled content which has to be hosted on peers possibly not trusted by all the participating content providers. Such an index is essential to improve the search performance of P2P access control aware data management systems. Yet another form of privacy breach can happen when individual users of a content publishing system host access controlled content on a third party provider who enforces access control on their behalf. In such cases, the sensitive data hosted is completely visible to the provider. The provider's infrastructure, with all its cumbersome privacy policies, may become a black box to the users who do not have a trustworthy auditing of their data accesses. The providers may expose the content to outsiders accidentally or purposefully often without the knowledge of the users. A classical example of such a case is online social networking, an emerging Web 2.0 paradigm. Customized privacy preserving P2P access controlled infrastructures that allow users to share their sensitive social profile data without the need of a big brother like Facebook[1] or Google[2] are becoming vital for building privacy aware online social networking. Yet they have to match the capabilities offered by the very expensive globally distributed data centers of the conventional social network service providers, which is a huge research challenge.

[1] www.facebook.com
[2] www.google.com

In this doctoral thesis, we propose mechanisms to build privacy preserving P2P systems to publish and share access controlled content by the participating peers. The thesis is composed, broadly, of the following main parts with dedicated contributions.

1. ***ACPeer- Access control aware P2P data management system*** is an access control aware structured P2P system which allows the publishers to exercise access control on the resources they publish. We exploit mutual trust among publishers to build a viable alternative for enforcing access control in P2P environment, a problem rarely addressed in the literature. The thesis explores the solution space elaborately. First, a naive mechanism of constructing independent networks is briefly presented followed by more efficient approaches that exercise access control either at querying time or at query response time. The thesis also describes a prototype implementation of one of these mechanisms.
2. ***PANACEA- PrivAcy preserviNg Access-ControllEd P2P system*** aims to improve the search efficiency of *ACPeer* systems. It proposes a novel tunable privacy- preserving indexing mechanism for access controlled content. The resulting index greatly reduces the search overhead. It can be safely hosted on any untrusted peers. In this system, privacy has two aspects: the deducibility of a resource's existence/non-existence and the discovery of the provider of the resource by an adversary. As proved both analytically and experimentally, the PANACEA system offers privacy closer to that of the best privacy- providing alternatives i.e., unstructured P2P systems and the search efficiency closer to that of the most search efficient alternatives i.e., structured P2P systems.
3. ***My3- Online Social Networking for privacy conscious users:*** Concentration of vast amount of personal information within a single administrative domain causes critical privacy concerns. As a result, privacy-conscious users feel disempowered with today's online social networks. *My3* is a privacy-aware alternative for online social networking built using the P2P paradigm. The number 3 in *My3* represents that it is a 3^{rd} alternative for online social networking. The first alternative is the current mode of centralized social networking like Facebook, Google Plus[3] while the second being the fully decentralized approach employing always-on personalized servers like Diaspora[4]. *My3* acts as a third alternative which builds a decentralized OSN bay replication schemes that exploit overlaps in online times of trusted friends who need not be always

[3] www.plus.google.com
[4] http://diasporaproject.org/

online. *My3* exploits, in its design, unique characteristics of current OSN workloads namely, user online activity patterns and locality of content accesses. The trust relationships among friends on the social network are considered while choosing replication points for the social network content. We propose different performance evaluation metrics for decentralized OSNs, namely, availability, availability-on-demand, update propagation delay, replication degree, access cost, and replication load. *My3* replication algorithms optimize one or more of these performance metrics. These algorithms take a user's set of trusted friends and their online times as input, and output a set of replication points. We evaluate *My3*'s performance using real world data traces from Facebook and Twitter. As experimentally found, high availability is achievable with a tolerable profile replication factor.

The thesis addresses the critical problem of sharing access-controlled content in a privacy preserving manner in a decentralized publishing environment. It proposes a novel indexing mechanism that improves the search efficiency of such systems. Finally, it demonstrates an application of such a system for the context of privacy-aware online social networking.

Keywords: *peer-to-peer systems, access control, privacy, privacy preserving indexing, online social networks, privacy in online social networks, decentralized online social networks.*

Résumé

Le paradigme peer-to-peer (P2P) est de plus en plus utilisé pour organiser les ressources de données distribuées pour diverses applications, comme par exemple l'édition et la distribution de contenu, les réseaux de stockage ouvert et les réseaux sociaux en ligne. En outre, les contenus à contrôle d'accès privé et semi-privé sur le réseau ont connu une croissance rapide ces dernières années, et plus particuliérement le contenu généré par les utilisateurs grâce à l'explosion des technologies du Web 2.0. Cependant, les systèmes P2P classiques de gestion de données n'ont pas réussi à suivre le rythme pour fournir des recherches efficaces sur du contenu à accès contrôlé par exemple, similairement à celle des systèmes P2P structurés. Par conséquent, leur adoption est limitée par le manque de possibilité de contrôler l'accès aux ressources partagées. En outre, dans des environnements ouverts avec des pairs non fiables, même lorsque des mécanismes adéquats restreignent l'accès aux ressources, des questions de confidentialité peuvent survenir en fonction de l'application, par exemple, une application pour construire un index de contenu à accès contrôlé qui doit être hébergé sur des pairs qui n'ont pas la confiance de tous les fournisseurs de contenu participant. Un tel index est essentiel pour améliorer les performances de recherche de systèmes de gestion de données P2P tenant compte des contrôles d'accès. Une autre forme de violation de la confidentialité peut se produire lorsque les utilisateurs d'un système de publication de contenu hébergent du contenu à accès contrôlé chez un fournisseur tiers qui applique le contrôle d'accés à leur place. Dans de tels cas, les données sensibles, ainsi hébergées, sont complètement visibles pour le fournisseur. L' infrastructure du fournisseur, avec toutes ses encombrantes politiques de confidentialité, peut devenir une bote noire pour les utilisateurs qui ne disposent pas d'un contrôle fiable de leur accès aux données. Les fournisseurs peuvent exposer le contenu à des tiers accidentellement ou intentionnellement, mais souvent l'insu des utilisateurs. Un exemple classique d'un tel cas est le réseautage social en ligne, un nouveau paradigme du Web 2.0. Les infrastructures P2P à accès contrôlé, sur mesure et préservant la vie privée qui permettent aux utilisateurs de partager leurs données de profil sociaux sensibles sans avoir besoin d'un "Big Brother" comme Facebook (www.facebook.com) ou Google (www.google.com) sont de

plus en plus essentielles pour la construction réseaux sociaux en ligne respectant la vie privée. Pourtant, ils doivent correspondre aux capacités offertes par les très coûteux centres de données distribués des fournisseurs de réseaux sociaux traditionnels, ce qui est un énorme défi de recherche.

Dans cette thèse de doctorat, nous proposons des mécanismes pour construire des systèmes P2P préservant la vie privée pour publier et partager du contenu à accès contrôlée par les pairs participants. La thèse se compose, au sens large, des parties principales suivantes avec leurs contributions spécifiques.

1. ***ACPeer-Système de gestion de données P2P conscient des contrôles daccès*** est système P2P structuré conscient des contrôles daccès qui permet aux éditeurs d'exercer le contrôle d'accès sur les ressources qu'ils publient. Nous exploitons la confiance mutuelle entre les éditeurs pour construire une alternative viable pour l'application de contrôle d'accès dans un environnement P2P, un problème rarement abordé dans la littérature. La thèse explore minutieusement l'espace des solutions. Tout d'abord, un mécanisme naf de construction de réseaux indépendants est brièvement présenté suivi par des approches plus efficaces qui appliquent le contrôle d'accès, soit au moment des requtes, soit au moment de la réponse à la requte. La thèse décrit également une implémentation dun prototype de l'un de ces mécanismes.
2. ***PANACEA-PrivAcy preserviNg Access-ControllEd P2P system*** vise à améliorer l'efficacité de la recherche dans les systèmes ACPeer. Il propose un nouveau mécanisme d'indexation adaptable respectant la vie privée pour du contenu à accès contrôlé. L'index qui en résulte permet de réduire considérablement les coûts de recherche et il peut tre raisonnablement hébergé sur nimporte quel pair non fiable. Dans ce système, la vie privée comporte deux aspects : la déductibilité de l'existence ou non d'une ressource et la découverte du fournisseur de la ressource par un adversaire. Tel que démontré à la fois analytiquement et expérimentalement, le système *PANACEA* offre une intimité plus proche de celle des meilleurs alternatives protégeant la vie privée, à savoir les systémes P2P non structurés et une efficacité de recherche proche de celle des alternatives les plus efficaces en terme de recherche, à savoir les systèmes P2P structurés.
3. ***My3-Réseau social en ligne pour les utilisateurs soucieux de leur vie privée*** : La concentration de grandes quantités de renseignements personnels dans un seul domaine administratif provoque des problèmes de confidentialité critiques. En conséquence, les utilisateurs soucieux de leur vie privée se sentent désemparés avec les réseaux sociaux daujourdhui. *My3* est un réseau social en ligne alternatif respectant la

vie privée construit en utilisant le paradigme P2P. Le numéro 3 dans *My3* indique qu'il sagit de la 3éme alternative de réseaux sociaux en ligne. La première alternative est le mode actuel de réseautage social centralisé comme Facebook, Google Plus (www.plus.google.com) tandis que la seconde est une approche totalement décentralisée employant des serveurs personnalisés, actifs en permanence, comme Diaspora (http://diasporaproject.org/). *My3* agit comme une troisième alternative qui construit une sur-couche réseau structurée et décentralisée avec des systèmes de réplication qui exploitent les chevauchements de temps en ligne entre amis de confiance qui ne doivent pas tre toujours en ligne. *My3* exploits, dans sa conception, les caractéristiques uniques des charges de travail actuelles OSN à savoir, les habitudes des utilisateurs activité en ligne et la localité d'accès au contenu. Les relations de confiance entre amis sur le réseau social sont considérés tout en choisissant des points de réplication pour le contenu des réseaux sociaux. Nous proposons différents paramètres d'évaluation de performance des décentralisées, à savoir, la disponibilité, la disponibilité à la demande, le délai de propagation mise à jour, le degré de réplication, le coût d'accès, et la charge de la réplication. *My3* algorithmes de réplication d'optimiser un ou plusieurs de ces indicateurs de performance. Ces algorithmes prennent ensemble un utilisateur d'amis de confiance et leurs temps de ligne comme entrée, et de sortie d'un ensemble de points de réplication. Nous évaluons les performances *My3* l'aide de véritables traces de données mondiales de Facebook et Twitter. comme expérimentalement trouvée, la haute disponibilité est réalisable avec un facteur de réplication profil tolérable.

La thèse aborde le problème crucial de partager l'accès est contrlé contenu d'une manière préservant la vie privée dans un environnement de publication décentralisée. Il propose un mécanisme d'indexation roman qui améliore l'efficacité de la recherche de ces systèmes. Enfin, il montre une application d'un tel système dans le contexte de la vie privée conscient de réseautage social en ligne.

Mots clés : *systémes peer-to-peer, contrôle d'accès, confidentialité, indexation respectant la vie privée, réseaux sociaux en ligne, la vie privée dans les réseaux sociaux, réseaux sociaux décentralisées en ligne.*

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Introduction

Chapter 1

Introduction

The last thing one discovers in composing a work is what to put first.

Blaise Pascal

1.1 Background

The peer-to-peer architecture was a revolutionary paradigm introduced when the client-server model was the only predominant architecture for information systems. It suddenly opened up vast possibilities for numerous novel applications that would have been unrealizable otherwise. For example, Bittorrent [2] made distribution of very large content at a fraction of cost, the traditional distribution mechanisms incur in. The $(key, value)$ data structure pursued as part of structured P2P systems, is widely used in a number of massively scalable data management systems that can live up to the requirements of today's storage demands [3, 47, 54]. The structured P2P architectures are increasingly employed for organizing distributed data resources for various applications, e.g., content publishing and distribution, open storage grid, and online social networking [45, 52, 89, 128]. On the other hand, explosive adoption of Web 2.0 technologies turned once-passive content consumers into hyper active content producers. There was more than 1200 billion giga bytes of data produced in the year of 2006 alone [24]. As a result, private and semi-private access controlled content on the network has grown rapidly in recent years particularly the user-generated content produced using Web 2.0 technologies.

However, the conventional P2P data management systems have failed to keep pace to support access controlled content. Hence, the adoption of the P2P systems is limited by the lack of possibility to control the access on the resources shared [49]. In any access controlled system, the access control policy defined by participating content providers must be enforced in a trusted manner. The entity which performs this referred to in the literature as a *reference monitor* [109]. In the case of P2P environment, its not feasible to identify a single point in the network to implement the reference monitors of all the

providers as there exists no single entity trusted mutually by all the participating peers. There exists a wide spectrum of possibilities regarding its positioning in the network: for example, at one extreme of the spectrum, peers individually implement it locally which restricts accesses to their content and at the other extreme, a single reference monitor supervises access to the entire content in the network. In between, a more viable solution exists which is flexible in a number of ways as discussed later in the thesis. Exploiting inherent trust among peers in a P2P system lead to more practical access controlled P2P systems [95].

In addition to enforcing access control on the content, maintaining the search efficiency on par with that of the structured P2P systems is a huge research challenge. Moreover, in open environments with untrusted peers, even when proper access control mechanisms are in place to restrict the access to the data resources, privacy issues may arise depending on the application. For example, in order to improve the search efficiency, building an index of the access controlled content is vital. This index has to be hosted on peers possibly not trusted by all the participating content providers. Peers search for the content through this index instead of broadcasting in the network without index, which greatly improves the search performance.

There is ever increasing need to design efficient privacy preserving indexing of the access controlled content. The privacy aspect generally considered in the literature is the *provider privacy* [42] where an unauthorized user should not be able to deduce with certainty which providers are sharing a document d in the network. For example, a conventional structured P2P system (e.g., Chord [120]) does not preserve any provider privacy as the *value* field directly reveals the exact provider information to an adversary. In a Chord- based P2P system, typically the *key* points to a data resource identifier and the *value* field points to a peer identifier where the data resource is hosted.

Yet another form of privacy breach can happen when individual users of a content publishing system host access controlled content on a third party provider who enforces access control on the users behalf. This is typically done when the 3^{rd} party provider offers infrastructure and services which the participating publishers can not procure themselves. In such cases, the sensitive data hosted is completely visible to the provider. A classical example of such a case is online social networking [84], an emerging Web 2.0 paradigm. The provider's infrastructure, with all its cumbersome privacy policies [32], may become a black box to the users who do not have a trustworthy auditing of their data accesses. The providers may expose the content to outsiders accidentally or purposefully often without the knowledge of the users. The privacy objectives of the users usually conflict with the business goals of the OSN providers which exploit the users sensitive data (for targeted advertising) to afford the infrastructural costs. The unprecedented proliferation of online social networking applications as shown in Figure 1.1 aggravated concerns of privacy advocates. Some times, an OSN provider may go out of business [4, 5, 6] and the user content used to reside on the provider's infrastructure may undergo serious privacy breaches which go largely unnoticed. Hence, in order to build privacy

Figure 1.1: Proliferation of Online Social Networks.

aware online social networking solutions, the research community proposed a number of approaches. Customized privacy preserving P2P access controlled infrastructures that allow users to share their sensitive social profile data without the need of a big brother like Facebook or Google are gaining the ground to be promising alternatives for privacy aware online social networking. They have to, yet, match the expensive globally distributed data centers of the conventional social network service providers in several efficiency and performance measures, which is a huge research challenge.

1.2 Contributions

In order to realize an efficient privacy preserving access controlled P2P data management system that has several applications, we addressed the following research challenges in this thesis work.

1.2.1 Access control aware P2P data management systems

The first contribution of this thesis is to emphasize the need for access controlled P2P systems and identify the efficient solutions to implement the reference monitors necessary to enforce the access control policies. We call an access control aware P2P data management system as *ACPeer* and propose overlay architectures for *ACPeer* systems. We exploit mutual trust among publishers to build a viable alternative for enforcing access control in P2P environment, a problem barely addressed in the literature. The thesis explores the solution space elaborately. First, a naive mechanism of constructing independent networks is briefly presented followed by more efficient approaches that exercise access control either at querying time or at query response time. The thesis also describes a prototype implementation of one of these mechanisms.

- *Access control in structured P2P systems:* We highlight the need for access controlled P2P systems especially structured P2P systems. We illustrate in detail the

types of access control feasible in such systems in terms of the granularity of the access controlled content. The resulting *ACPeer* systems are labeled as *Level-0* to *Level-2* based on the level of access control the corresponding system offers.

- *ACPeer system designs:* We propose solutions for enforcing access control in P2P systems which vary in terms of the network locations where the reference monitor is implemented. In a *query*-time access control enforcement, the reference monitor is pushed into the network and where as, in *reply*-time access control enforcement, it exists with the hosts where the access controlled content resides.

- *ACPeer prototype implementation:* One of the *ACPeer* designs, namely the query-time access control realization is implemented as part of the TEAM project [34] which is elaborated in corresponding chapter.

1.2.2 Privacy preserving indexing of access controlled content

The thesis addresses the issue of search efficiency in *ACPeer* systems and adopts indexing as the way for improving the search performance. But since the index will be often hosted on untrusted peers in the network, we argue that such an index should preserve the privacy of the access controlled content.

- *Resource and provider privacy:* In addition to the *provider privacy* usually addressed in the literature [42] for building privacy preserving indexing, the thesis argues for the need of *resource privacy* as well. The former prohibits an adversary from identifying a provider precisely for an arbitrary resource he/she is interested in. The latter privacy aspect prevents the adversary from even deducing whether the resource is present or not in the system with any of the participating providers. As proved in the corresponding chapter, resource privacy complements the provider privacy significantly in realizing a system offering improved privacy over the existing systems.

- *PANACEA- PrivAcy preserviNg Access-ControllEd P2P system*: We propose the PANACEA system which employs a novel tunable privacy- preserving indexing mechanism for access controlled content. The resulting index greatly reduces the search overhead and it can be safely hosted on any untrusted peers. The index generated offers both resource and provider privacy of the access controlled content. As proved both analytically and experimentally, the PANACEA system offers privacy closer to that of the best privacy-providing alternatives i.e., unstructured P2P systems and the search efficiency closer to that of the most search efficient ones i.e., structured P2P systems.

- *Privacy analysis methods:* We propose novel methods to analyze privacy offered by such systems using both probabilistic and information theoretic approaches.

1.2.3 Privacy aware Online Social Networking

Unprecedented aggregation of massive personal information within a single administrative domain in today's online social networks makes privacy advocates and government agencies worrisome alike. This thesis addresses the issue of designing privacy-aware alternative for online social networking and proposes solutions built using the *ACPeer* systems.

- *Feasibility study of decentralized OSNs:* The thesis deals with the emerging issue of decentralized social networks and studies empirically, the question of *Is decentralized social networking feasible?* For the decentralized social network paradigm to become a serious alternative to the current centralized infrastructure based OSNs, some key design objectives, often conflicting with each other, have to be met. In this thesis, we explore such design objectives concerning various system properties, namely availability, replication degree, user online times, privacy offered, and experimentally study the trade-offs among them based on real data sets from Facebook and Twitter[1]. We introduce different mechanisms to model user online behavior in the OSN from their OSN activity times. We demonstrate how different profile replica selection approaches significantly affect the system performance. We argue that a decentralized social network based on friend-to-friend networks is a more viable solution for privacy preservation.

- *My3- Online Social Networking for privacy conscious users:* We propose *My3* a privacy- friendly decentralized online social network. The *My3* system makes use of the trust among friends in a social network for building a highly available storage for decentralized social network. The number 3 in *My3* represents that it is a 3^{rd} alternative for online social networking. The first alternative is the current mode of centralized social networking like Facebook, Google Plus while the second being the fully decentralized approach employing always-on personalized servers like Diaspora. *My3* acts as a third alternative which builds a decentralized OSN by replication schemes that exploit overlaps in online times of trusted friends who need not be always online. The *My3* system exploits the well-known interesting properties of the current online social networks in its novel design namely, locality of access, predictable access times, geolocalization of friends, unique access requirements of the social content, and implicit trust among friends. It allows users to exercise finer granular access control on the content, thus making *My3* extremely privacy-preserving. Moreover, we propose different replication strategies that users may independently choose for meeting their personalized performance objectives. A detailed performance study evaluates the system regarding users' OSN profile content availability, access delay, freshness, and storage load. By using real-world data traces, we prove that *My3* offers high availability (close to 1) even with low

[1] www.twitter.com

average per-user online time (in the order of tens of minutes per day) in the network.

1.3 Thesis Organization

The remainder of the thesis is organized as follows: Chapter 2 discusses the related work in the literature corresponding to the research issues the thesis has addressed. Chapter 3 deals with the problem of access controlled P2P systems and provides several design choices including their qualitative evaluation. In Chapter 4, the PANACEA system is introduced. A detailed evaluation of the system over existing state-of-the-art is provided. The issue of decentralized privacy preserving online social networks is treated in Chapter 5. Design objectives of such systems are identified followed by several replication algorithms and their performance evaluation using real-world Facebook and Twitter datasets. In Chapter 6, the *My3* system is detailed and its effectiveness as a privacy aware online social network is illustrated with detailed performance evaluation using the above datasets. We conclude the thesis work in Chapter 7 and lay out detailed future work.

1.4 Selected Publications

This thesis is based on the following main publications:

- N. Rammohan, T. G. Papaioannou, and K. Aberer. A Decentralized Online Social Network with Efficient User-Driven Replication. *IEEE International Conference on Social Computing (SocialCom 2012)*, Amsterdam, 2012. [Acceptance Rate: <10%]

- N. Rammohan, T. G. Papaioannou, and K. Aberer. Towards the Realization of Decentralized Online Social Networks: an Empirical Study. *IEEE ICDCS International Workshop on Hot Topics in Peer-to-Peer Computing and Online Social Networking (HotPost)*, Macau, China, 2012.

- N. Rammohan, T. G. Papaioannou, and K. Aberer. My3: A Highly-available P2P-based Online Social Network. *11th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'11)*, Kyoto, Japan, 2011.

- N. Rammohan, T. G. Papaioannou, and K. Aberer. Privacy-aware and Highly-available OSN Profiles. *19th IEEE International Workshops on Enabling Technologies - Infrastructure for Collaborative Enterprises (WETICE 2010)*, 2010.

- N. Rammohan, T. G. Papaioannou, Z. Miklos, and K. Aberer. Tunable Privacy for Access Controlled Data in Peer-to-Peer Systems. *22nd International Teletraffic Congress (ITC 22)*, Amsterdam, 2010.

- N. Rammohan, Z. Miklos, and K. Aberer. Towards Access Control aware P2P Data Management Systems. *EDBT International Workshop on Data Management in Peer-to-Peer Systems*, St Petersburg, 2009.

The following work is also part of this thesis.

- N. Rammohan, T. G. Papaioannou, and K. Aberer. Privacy Preserving Indexing of Access Controlled Data in Peer-to-Peer Systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE), under preparation..*

Although the following publications are not directly related to this thesis, it helped to understand the significance of privacy preserving mechanisms in other practical domains, such as, participatory sensing.

- B. Agir, T. G. Papaioannou, N. Rammohan, K. Aberer, and J.-P. Hubaux. Adaptive Personalized Privacy in Participatory Sensing. *5th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, Tucson, Arizona, USA, 2012.

- B. Agir, T. G. Papaioannou, N. Rammohan, K. Aberer, and J.-P. Hubaux. User-side Adaptive Protection of Location Privacy in Participatory Sensing. *Submitted to 11th International Conference on Mobile Systems, Applications and Services (MobiSys 2013).*

Chapter 2

Background and State of the Art

If I have seen further it is by standing on the shoulders of giants.

Isaac Newton

2.1 Introduction

In this chapter, we review in detail, the work in the literature related to this thesis, in particular the literature on the conventional peer-to-peer systems, their corresponding access control issues and solutions, and privacy preserving sharing of access controlled content in a multi-publisher environment. In addition, we deal with privacy issues in online social networking, an emerging Web 2.0 paradigm and how research community resorted to decentralized alternatives that enable sharing of access controlled social content in a privacy preserving manner.

This chapter is organized as follows: In Section 2.2, a general introduction to the conventional peer-to-peer systems is provided along with the functioning of routing in different overlay networks. In Section 2.3, an overview of access control in information systems is provided followed by a greater discussion of access control issues in structured P2P systems. We also provide the necessary background for the contributions we made in Chapter 3, where we propose mechanisms for enforcing access control in P2P systems. In Section 2.4, we discuss how search efficiency in such systems can be improved by using privacy preserving indexing of the access controlled content. We present the relevant work in the literature corresponding to this problem which we addressed in detail in Chapter 4. In Section 2.5, we address the emerging research topic of privacy issues in online social networks and discuss briefly how it is addressed in the literature. We will also provide a high level overview of access controlled P2P systems evolving as a strong alternative for privacy preserving online social networks which we deal with, in detail in Chapter 6.

2.2 Peer-to-Peer Systems

The peer-to-peer communication model has introduced a significant paradigm shift in the way users share data resources and communicate with each other on the Internet. The peer-to-peer overlay architectures have undergone several redesigns over the years. The fall of the first partially centralized peer-to-peer file-sharing systems (Napster [29]) led to the creation of the second generation of peer-to-peer overlays (Gnutella [26], Freenet [48]). The simple protocols and the unstructured, decentralized nature made these networks robust and free from drawbacks of the centralized approaches. However, later it had soon become evident that the ad-hoc architecture of Gnutella-like systems does not scale and uses excessive amounts of network bandwidth. In unstructured overlays such as Gnutella, there are no constraints on how the topology can form as peers arrive and depart from the system. The peers that other peers can connect to, are either discovered through random walks or flooding as shown later. More efficient, structured overlays appeared (P-Grid [37], Chord [120], Kademlia [89]), which used the existing network and computing resources more effectively. Unlike their unstructured counterparts, the topology of the structured overlays is not arbitrary. Link establishment among the peers is usually strictly defined by specific protocols. The topologies can result in various regular structures like rings [120], tries [37], multidimensional tori [102], de Bruijn graphs [60]. Structured P2P systems based on Distributed Hash Table (DHT) paradigm are increasingly adopted in building massively scalable and highly available data management systems. The academic research community has pursued a wide range of such systems which include- Chord [120], Pastry [106], P-Grid [37], and CAN [102]. DHT based scalable storage infrastructures are embraced beyond the realms of academia also [36, 54, 128].

Because of the promising characteristics such as massive scalability, zero administration overhead, low cost, and autonomous control, the structured P2P systems are ideal for realizing collaborative networked systems built from resources shared and owned by cooperative groups of users. Such systems demand high degree of autonomicity and lesser administration overhead, and the P2P infrastructures are an ideal choice for the said requirements. Cooperative File System (CFS) [52] reports an efficient robust file system realized with a completely DHT based decentralized architecture. CFS achieves the performance of traditional FTP server based file systems, and promises a high robustness to node failures and high availability of the data using replication. A cooperative backup system is proposed in [83], where a set of hosts form a cooperative P2P network and backup the data of each other. Each participating computer is assigned to a small set of geographically distributed computers which share their storage to backup the data of the computer. Thus they achieve a cooperative backup scheme which is orders of magnitude cheaper over traditional commercial tape based backup schemes. In addition, FARSITE is another system that [39] aims at a serverless, secure, and scalable file system targeted

for working environments involving a few thousand hosts typical of a corporate working environment.

Efforts are made to build distributed semantic storage infrastructures on top of P2P cooperative storage. The NEPOMUK project [30] uses a P2P semantic storage system to extend the personal computer into a collaborative environment and improves the state of art in online collaboration and personal data management. The TEAM project [34] creates a collaborative P2P semantic infrastructure for lightweight knowledge sharing optimized for distributed software teams.

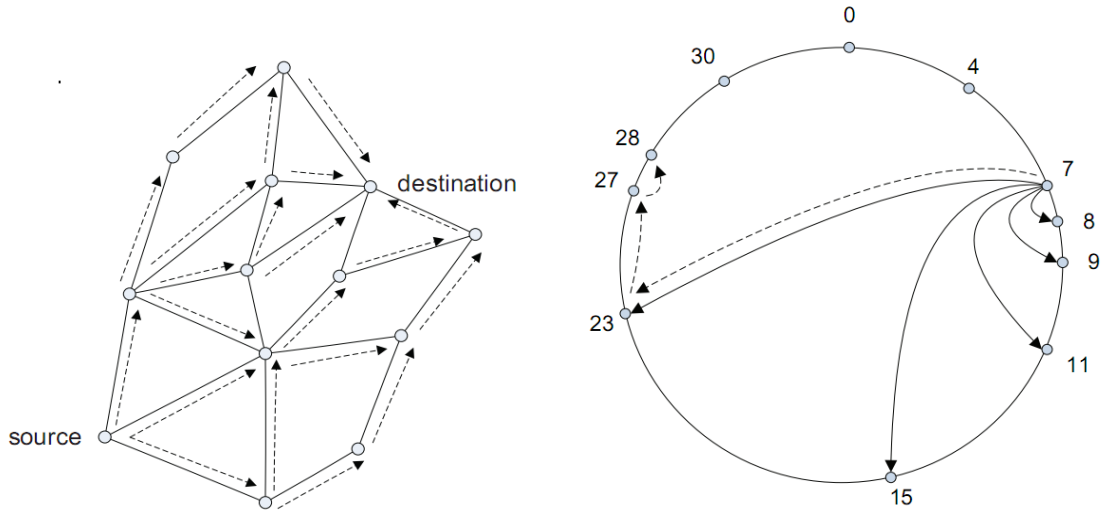
At the same time, the W3C consortium is actively pushing the semantic technology based information systems towards the next generation Internet. A number of technologies such as Resource Description Framework (RDF), RDF-Schema (RDFS), and Web Ontology Language (OWL) are already standardized by the respective working groups of the organization [72]. Semantic information systems based on peer-to-peer infrastructures are gaining huge importance as the peer-to-peer communication model can better handle the semantic heterogeneity of such systems. GridVine [38] system builds a P2P semantic RDF store which handles the semantic heterogeneity using schema mappings inserted by peers. A query on the network automatically gets reformulated using these mappings. There are a number of P2P based RDF triple stores proposed in the literature [46, 75].

2.2.1 Routing in P2P Systems

Peer-to-peer overlay networks enable the peers to communicate with one another even if the communicating peers do not know their physical addresses in the underlying network, e.g., the IP address on the Internet. The way it is achieved in the overlays is by routing overlay messages. Each overlay message originates at a source and is forwarded by the peers in the overlay until the message reaches one or more destinations. A number of routing schemes have been proposed with each routing technique dictating the P2P overlay characteristics.

2.2.1.1 Unstructured P2P systems

Unstructured overlay networks use mainly two mechanisms to deliver routed messages: flooding and random walks. When some peer v receives a flood message from one of its overlay neighbors w, then v forwards the message to all of its neighbors except w as shown in Figure 2.1a. When v receives the same message again, it is ignored. Eventually the flood reaches all of the destinations. For example, in Gnutella [26], a file-sharing peer-to-peer system, a peer s that wants to download a file floods the network with queries. If some peer d that has the file desired by s is reached by the query flood, then d sends a response back to s. Flooding consumes a significant amount of network bandwidth. To reduce it, the flooded messages typically contain a Time-To-Live (TTL) counter included in every message. The counter is decremented whenever the message is forwarded. This limits how far the flood can spread from the source but at the same



(a) Flooding in an Unstructured P2P System. (b) Routing in a Structured P2P System (Chord).

Figure 2.1: Routing in P2P Systems

time lowers the probability of reaching the peer that holds the searched file. The high bandwidth usage of flooding has led to the design of an alternative routing scheme for unstructured overlay networks: i.e., random walks [65]. Instead of forwarding a message to all of the neighbors, it is only forwarded to a randomly chosen one. Depending on the network topology random walks provide different guarantees of locating the destination peer, however all of the random walk approaches share one disadvantage: a significant and, in most cases, intolerable latency.

2.2.1.2 Structured P2P systems

As more peers join the overlay network and as there are more messages that need to be routed, flooding and random walks quickly reach their scalability limits. This problem has prompted the research on structured overlays. In structured overlays, each peer has a unique and unchanging identifier picked when a peer joins the overlay. The peer identifiers enable efficient routing in the structured overlays. Each routed message has a destination identifier to which the message needs to be delivered. Instead of blindly forwarding the message to all neighbors, as in the unstructured overlays, each peer in a structured overlay forwards the message to only one neighbor in each hop. The next hop is selected by using the concept of peer identifier distance. Most of the modern structured overlays define the notion of distance between any two peer identifiers. For example, in Chord [120], identifiers are selected from the set of integers $[0, 2m-1]$ and are ordered in a modulo $2m$ circle. The distance $d(x, y)$ between two identifiers x and y is defined as the difference between x and y on that identifier circle, i.e., $d(x, y) = (y - x) mod 2m$.

The routing in the Chord system is illustrated in Figure 2.1b. The big circle represents the peer identifier space with IDs in the interval $[0, 31]$. The small circles are

the peers and the number beside them is their ID. Peer 7 is connected (solid arrows) to peers with exponentially increasing distance from 7: $7+1=8$, $7+2=9$, $7+4=11$, $7+8=15$, $7+16=23$. Assume that peer 7 wants to route a message to peer 28. Dashed arrows represent the routing path. The peer 23 is the neighbor of 7 that is closest on the ring to the destination 28 and that neighbor is chosen as the first hop for the message. One hop is not enough to reach 28, but the peer 23 brings the message closer to the destination and peer 27 finally delivers it. The greedy routing rule of always selecting such next hop that brings the message as close as possible to its destination is the main building block of all structured overlay networks.

In another overlay, Kademlia [89], the identifiers are 160-bit integers and the distance between two identifiers x and y is defined as their exclusive bitwise OR (XOR) interpreted as an integer, i.e., $d(x,y) = x \oplus y$. Although the modern structured overlays differ in the details of how they make use of the peer identifiers for routing, they are all based on the same general greedy routing principle. When some peer v receives a message with a given destination identifier, it forwards the message to that next hop whose identifier is the closest to the destination identifier. In other words, in every hop, the message gets as close as possible to the destination. Routing terminates when Time-To-Live of the message is exhausted or the peer is the message destination. The topology of most structured overlays is designed to be scalable, that is, all destinations can be reached from any node in a logarithmic number of hops.

2.3 Access Control in P2P Information Systems

2.3.1 Motivation

Private and semi-private content on the network has grown rapidly in recent years particularly user-generated content thanks to the explosion of the Web 2.0 technologies. However, the conventional P2P data management systems have failed to keep pace to support access controlled content with search efficiency similar to that of the structured P2P systems. In contrary to the centralized systems and the unstructured P2P systems which have no indexing hosted, there exist more than one point in a structured P2P network to enforce access control. Hence, designing access controlled structured P2P systems poses several design challenges which was ignored to a large extent by the research community so far.

2.3.2 Access control overview

Any access control system in the context of an information system guards accesses on the information/data shared in the system which is referred as a *resource* in general. A resource can be a file, an RDF triple or any data object. In any access control model, the entities that can perform actions in the system are called *subjects* or *principals*, and the entities representing resources to which access may need to be controlled are called *objects*. Set of operations a subject can perform on an object are usually represented

in an *access control matrix*. Typical examples of such operations in the context of information systems are read, write, update, and delete [109]. A subject/principal can be a single or a group of users in a system. For example, the following access control policy statement typically used in UNIX installations, states that subject *rammohan*- an individual user and subject *lsir*- a group of users can perform *read* and *write* operations on an object/resource *thesis.tex*. The subject "rammohan" is said to be *authorized* to "read" the resource "thesis.tex".

```
-rw-rw-r-- 1 rammohan lsir 6214 Aug 16 12:46 thesis.tex
```

Access control models used by current information systems tend to fall into one of two classes: those based on *capabilities* and those based on *access control lists (ACLs)*. In a *capability-based* access control, mere posessing a capability grants access to the resource in question. For example, a valid decryption key grants access to a resource encrypted with the same key. In this model, an access can be granted to a subject by passing the corresponding capability through some communication channel. In an *ACL-based* access control, a subject's access to an object depends on whether its identity is on a list associated with the object, for example, a user's photos on Facebook accessible to only his friends. Access in such systems is granted by editing the list, per say, by the owner of the photos.

2.3.2.1 Access control models

There exist a number of ways how subjects are represented in an information system and the extent to which access control is enforced. These are usually specified in an *access control model* which are most widely categorized as [109]:

1. Discretionary Access Control (DAC) model
2. Mandatory Access Control (MAC) model
3. Role Based Access Control (RBAC) model

In discretionary access control (DAC) model, an access control policy is specified by the owner (i.e., creator) of an object, for example, in the form of access control matrix. The management of access privileges is completely at the discretion of the owner only. When a subject tries to perform an operation on a data object, the subject is authorized to do so provided a corresponding rule exists in the policy. The entity which ensures this authorization is called as *reference monitor* which is explained later. The DAC model is the most widely adopted in commercial systems due to its simplicity and flexibility. However, in a DAC system, a user who has a legitimate access to a data resource, say read access to a file, can later redistribute the file to other users in the system not intended by the original owner.

A mandatory access control system (MAC), however, prevents such unattended access breaches. A MAC system allows access to a resource if and only if the access policy

explicitly mentions the same. MAC is used typically in military information applications. The system assigns certain security or privilege level to each subject and object in the system. A subject's security level specifies its level of trust, where as, that of an object specifies the level of trust required for access. In order to access a given object, the subject must have a security level equal to or higher than that of the requested object [109]. MAC strictly controls the information flow in the system and ensures that information from high level objects do not flow to low level subjects. In addition, a subject at lower privilege level is prevented from accessing resources configured with higher privilege.

However, neither the DAC nor the MAC sufficiently addresses the needs of the most of the commercial working environments. Hence, role-based access control (RBAC) is introduced, where an access policy is determined by the system, not the owner. It controls the access to the objects based on the type of activities (roles) users perform in the system. Authorization rules are specified in terms of roles subjects play in the system. For example, only *managers* can update salary records of *employees*. Role-based access control [109] provides a scalable alternative for access control lists, where privileges are configured for roles users posses. There is no need for configuring the privileges for each and every user in the system as done in the case of access control lists. In this thesis, we assume DAC based access control model as it is more relevant to an independent multi-provider data publishing and sharing environment.

Granularity of access control w.r.t both the subjects and the objects is inherently dependent on the kind of access control model deployed in the system. In RBAC, the granularity of subjects can not be finer than the level of roles. Assigning privileges to a single user is not possible unless a special unique role is created for that user alone. However, this is not the case with DAC, since it offers finer granularity up to an individual user. A per-resource access control is said to offer finer object granularity over a model which offers access control only at a collection of resources. For example, a system which allows either all or none of the contacts on a phone to be read by any application is *coarser* in nature compared to a system which restricts access to an individual contact in the contacts book. If the storage system does not allow flexible handling of granularity, it leads to data leakage to unauthorized subjects as observed in [116].

2.3.2.2 Reference monitor

A *reference monitor* is an entity (e.g., a software component) which enforces an access control on the subjects while performing operations (e.g., read and write) on objects (e.g., files) in a system. In any access controlled system, every access to an object by a subject must pass through the reference monitor. A truthful secured realization of a reference monitor is a challenge to any access controlled system. In a multi-publisher environment like P2P information system, the reference monitor may be distributed across the network. In structured P2P systems which build an index of the content

shared in the system, there exist various network points where the reference monitor can be implemented as will be illustrated later in the thesis.

2.3.3 Access control in P2P systems

The problem of controlling accesses on shared data is widely studied by the research community. Researchers proposed a number of access control models and techniques to enable access of the files or resources by only the intended users. An access control aware P2P information system (referred to as *ACPeer* in the following for convenience) can be assumed to be analogous to a widely-studied problem of hosting files or data on untrusted storage. Such systems use cryptographic techniques to realize access control aware data accesses. The Plutus [74] describes a file system that aggregates files with similar privileges into groups and encrypts all the files in the group with a single key. The key management and distribution lies in the user's hands and is done out-of-band. SiRiUS [66] describes a filesystem which provides access control in the untrusted storage model using public key cryptography for all metadata operations. The system is developed keeping no-modification-to-existing-file- server in consideration, yet achieving secured networked file system. Similar public key cryptography technique was used in FARSITE [39]. SUNDR [81] presents a network file system to store data securely on untrusted servers. Any violation of access privileges or unauthorized modifications by malicious users can be detected by the owners. The system is designed to achieve a property called *fork consistency*- clients can detect any consistency failures as long as they see each other's file modifications. The work was primarily motivated to design safe code repository hosting servers of few open-source and commercial projects as traditional centralized storage were observed to be vulnerable to malicious attacks. Most of these works target more at securely outsourcing storage to third party storage providers than the problem of access control. In [116], the authors address the problem of access control in detail and introduce a robust access control framework using cryptographic techniques. They improve on the existing access control model for UNIX-like systems and apply it to the accesses on data hosted on untrusted storage providers. However, the data access patterns in structured P2P systems vary from that of the these systems, where peers publish index for each resource shared. The searching peers first consult this index before accessing the original resources.

There is a little work done in the literature on access control mechanisms for structured P2P systems, whose presence is increasingly expanding into a number of different application domains. The work in [51] provides a policy based access control framework for P2P grid systems. An access control system for collaborative environments involving mobile and P2P systems is addressed in [58]. Access Control satisfiability is used in [125] to compute the trustworthiness of acquaintances in a P2P overlay network. Modeling access control in the case of P2P collaborative systems is addressed in [131]. The authors propose a fine granular and attribute based access control framework, where each peer assumes a group role and an application role. Then an access control policy which maps

various roles and permissions is configured and the underlying framework executes the policy. Our work is based on the lines of PHera [49] which proposes a scalable and fine-grained access control framework for P2P infrastructures. They deal with super-peer based P2P overlays where, sub peers specify their access control policy and the super peer enforces it on their behalf. Any invalid request for a resource will not cross the super peers and reach the peers. Policy statements of individual policies are grouped for scalability and performance reasons. However, this work assumes that all super peers are trustworthy to enforce the access control policy of the sub peers. In any case, the sub peers, before processing an access request, can still verify the access control decisions of super peers.

We consider the problem of designing access control aware P2P data management systems in Chapter 3. We discuss several mechanisms to embed realization of reference monitors into P2P overlay design. We argue that access control aware system should also enforce access control on the index hosted in structured P2P systems, a concern, overlooked by PHera which is targeted at super peer based P2P systems instead of structured P2P systems that we deal with.

2.4 Privacy Preserving Indexing for Search-efficient *ACPeer* Systems

2.4.1 Motivation

Private and semi-private content on the network has grown rapidly in recent years particularly the user-generated content thanks to the explosion of the Web 2.0 paradigm. However, the conventional P2P data publishing systems have failed to keep pace to support access controlled content with search efficiency similar to that of the structured P2P systems. A searcher for an access controlled document, typically has to identify and query each individual relevant provider for the document. This mandates that the searcher should remember all relevant repositories. Hence, an indexing mechanism which improves the search efficiency- for example, that can precisely narrow down set of repositories where, the document in search possibly can exist- is vital for access controlled P2P systems to be practical. We recognize this issue while designing an *ACPeer* system in Chapter 3 and the solution we propose in Chapter 4 greatly complements the *ACPeer* design.

As noted earlier, structured P2P systems host index of the form $(key, value)$ pairs in the DHT. Often this index is stored on untrusted peers as dictated by the routing protocol of the structured overlay. Note that there exists no trustworthy way to enforce access control in such untrusted environment on the index especially still keeping it searchable. However, if the same indexing techniques are used for structured *ACPeer* systems, such an index may leak certain information about the data resources to unauthorized users who could not have known without accessing the data resource itself. This type of information leakage is often termed in the literature as *privacy leakage* of the

access controlled data resources. Since index is vital for efficient search in structured P2P systems, one needs privacy preserving indexing for structured *ACPeer* systems. In Chapter 4, we deal with the problem of privacy preserving indexing of access controlled content for multi-publisher P2P systems. Encrypting the index will preserve the privacy, per say, results in key management issues and loss of searchability for a searcher without the encryption keys. Hence, only an encryption-free index would meet the objective.

In Chapter 4, we introduce the *PANACEA*- PrivAcy preserviNg Access-ControllEd P2P system, which is used to build a privacy preserving index of the access controlled content shared by participating providers/publishers. The resulting index can be hosted on untrusted server or peers in the P2P system. The privacy aspects considered are the privacy of both the access controlled data resource (e.g., file) and the corresponding provider, which are explained in detail in the corresponding chapter. PANACEA employs anonymization techniques for building the index.

2.4.2 Privacy preserving data publishing and retrieval

There is significant research work in the literature related to PANACEA, particularly in the areas of access control in P2P systems, privacy of access-controlled content, and anonymous P2P systems. The privacy preserving data publishing was widely addressed by the database community in the literature [62]. The most popular approach is the anonymization techniques that remove/blur the sensitive data from database records before releasing for publishing. In k- anonymization technique, an individual database record is stripped off personally identifiable information so that it can not be differentiated with at least k- other database records [62]. Other solutions include *l-diversity* [77], *t-closeness* [82], etc.

The authors in [40] envision a privacy preserving framework that makes information released unusable to illegitimate users. Privacy preserving full-document indexing is addressed in [130] which builds a distributed indexing infrastructure which allows faster search of the documents without leaking information about other documents in the system. The indexing mechanism offers tunable trade-offs between security and retrieval properties of the index.

The OneSwarm system proposed in [73] employs an unstructured friend-to-friend overlay for privacy preserving content sharing. It preserves the privacy of a peer's network location (IP address) using cryptographic mechanisms. The system allows users to define permissions for the data shared among trusted friends. Peers search for data objects using flooding techniques, similar to access-controlled unstructured systems.

The Crowds [103] and the Tor [35] systems offer anonymity for activity on the web using similar randomized forwarding techniques used by PANACEA. However, PANACEA improves the technique so as to be suitable for privacy preserving publishing as explained in the corresponding chapter. There exists a large number of works in the area of anonymous P2P systems that achieve publisher (source) or reader (searcher) anonymity or both [48, 117]. Additionally, the anonymity of a node hosting an index entry (resource)

is also considered [56]. In Freenet [48], resource identifiers are generated in several cryptographic ways and are inserted into the system based on these identifiers. It achieves access control, the resource and provider privacies using cryptographic techniques, which however, involve complicated key distribution and management overhead. P2P access control system based on such cryptographic indexing was discussed in [95].

A hybrid P2P system involving structured and unstructured topologies to achieve the sender and receiver anonymity, was discussed in [117] and referred to as Agyaat. By connecting unstructured sub-overlays via a DHT, sender and receiver identities remain hidden. Agyaat, provides mutual anonymity for the sender and receiver, which is not among the goals of PANACEA. Agyaat offers three alternative resource discovery approaches: semantic groups, centralized directory service, and dynamic services. In the first case, peers that host semantically similar resources are grouped into a cloud. Then, some sort of resource and provider privacies can be provided at the expense of resource discovery, which is flooding-based, as opposed to our approach. For improving resource discovery, a centralized directory service or dynamic services can be also employed. Then, a resource is mapped to a cloud and the index is stored at a central server or at the coordinator peers of the clouds in a distributed manner. These are similar to the privacy preserving indexing employed in [42] as explained below.

To enable access control in P2P systems, PHera [49] proposes a fine-grained access control framework based on super-peer-based P2P overlays where, the access-control policies of sub-peers are enforced by the super-peers. Super peers index the data of sub-peers and they could preserve data privacy by not replying to the queries from unauthorized peers. However, this approach assumes that all super-peers are unanimously trusted by their sub-peers to protect their data privacy and enforce the access control policies. Choosing a trusted peer mutually trusted by all sub-peers is very difficult [42].

2.4.3 Privacy preserving indexing of access controlled content

Regarding the specific problem of privacy of access-controlled content mainly addressed by PANACEA, a privacy-preserving approach for centralized indexing of such data is proposed in [42]. The authors argue that an adversary A should not be allowed to deduce that a provider p is sharing some document d unless A has been granted access to d by p. Hence, the index constructed should preserve the privacy of publishers which is termed as *provider privacy*. The extent to which an indexing mechanism preserves the provider privacy determines the degree of privacy. This work including PANACEA makes use of the degree of privacy achieved with the respective proposed mechanisms in terms of the following notions, originally developed in [103] and shown in Figure 2.2.

- *Provable Exposure:* The adversary can deduce with full certainty that p is sharing d.
- *Possible Innocence:* The claim of adversary about p sharing d can be false with a non-trivial probability (e.g., with probability in (0:5, 1)).

- *Probable Innocence:* The claim of adversary about p sharing d is more likely to be false than true (e.g., with probability in (0, 0:5]).
- *Absolute Privacy:* The adversary cannot determine if p is sharing d or not.

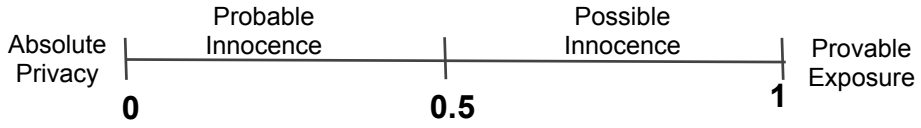


Figure 2.2: Degrees of privacy measured in terms of the probability that an adversary can deduce with certainty that a provider p published a document d.

In this terminology, the conventional search index agnostic to provider privacy allows a provider to be *provably exposed* through the index itself. The work in [42] builds a privacy preserving index which leaves providers with at least *probable innocence* in response to adversary attacks on the index i.e., the probability of deducing the provider by an adversary will be less than 0.5. The resulting index can be hosted on any untrusted server without the worry of breach of the provider privacy. The index is constructed using a randomized procedure which partitions providers into privacy groups of a fixed constant size c ($c > 2$). Within a group, providers are arranged in a ring. All providers in a ring iteratively circulate for a fixed number of rounds, a bloom filter representing the content hosted on the providers, bits of which are set probabilistically by the proposed algorithm which has a very small probability of error. At the end of this iterative process, the resulting bloom filter forms the *content vector* representing the index of the content that all the providers in the group own. After such iterative rounds in all the privacy groups, the content vectors of each group are sent to a designated centralized index aggregator. The aggregator also collects the list of all providers in each privacy group. It then aggregates these vectors into a final single index which maps a bit position i to a list of providers that belong to privacy groups whose content vector has i set to 1. Thus content (for example, a document d) mapped to bit position i is advertised as if owned by all the providers inside the group thus the corresponding original provider is blended into a crowd of size c. A searcher searching for the document contacts all the providers inside a group individually and queries for it. The access control is exercised before release of the document by the corresponding provider. Thus, in this model, the reference monitor is installed on individual providers itself guaranteeing the most truthful exercise of the access control policy. However, as opposed to PANACEA, the approach in [42] does not address the resource privacy which we introduce in Chapter 4. Furthermore, new resources can be easily inserted into the index of PANACEA, while index reconstruction is required in [42]. In addition, the provider privacy is breached in some scenarios in this approach, where a provider may suffer a breach larger than *probable innocence* from adversaries within his privacy group. As proved by the authors, providers who immediately precede an active adversary in the ring of a privacy group, will be assured of only *possible innocence*. Specifically, an adversary neighbor can determine whether

its predecessor along the ring is sharing a document with a probability higher than 0.5. In one another scenario, a provider p is *provably exposed* if a succeeding and preceding provider of p in the ring of providers collude, overcoming which involves encryption mechanisms.

2.5 *ACPeer* Systems for Privacy-aware Online Social Networking

2.5.1 Motivation

The online social network paradigm has taken the Web 2.0 penetration into unprecedented scales by offering innovative tools for networking among users and distributing the user-generated content to millions of interested users. The conventional social networks (e.g., Facebook, Google Plus) have recently seen an explosive growth. Facebook has currently nearly 900 million users active on the service at least once in a month. As a result, these OSNs have become store houses of unprecedented amount of data in the form of messages, photos, links, and personal information. Online social network portals continue to evolve towards one-stop hubs for content in a way that influences the future of the Internet [91]. However, social networking portals that are operated in a cloud infrastructure maintained by a single authority (e.g., Facebook or Google) will strategically have greater stakes in protecting the interests of the advertisers than addressing the privacy requirements of the users. During sign-up time, users consciously or unconsciously permit the organizations to share their personal information with third-parties in whatever form the organizations choose to [70]. In addition, the leakage of personal information from OSNs can be associated with the user activity on non-OSN sites as well [79]. Moreover, as social information owned by a single authority grows, so does its financial power. Even if we trust that the provider is motivated to protect the user data, large-scale data breaches are still possible. Data on the cloud is exposed to attacks that did not exist in traditional architectures, as seen in recent, high-profile exploits affecting major infrastructure providers [7, 8, 9].

As privacy concerns of today's online social networks dominate their utility, the access control aware structured P2P *ACPeer* systems can be strong alternatives which restore the control of their sensitive data back to the OSN users. We deal with this research challenge in Chapters 5 and 6. In the former, we scrutinize the feasibility of building decentralized online social networks based on *ACPeer* systems using real datasets from Facebook and Twitter. In the latter chapter, we introduce the *My3* system, a privacy-aware alternative for online social networking.

2.5.2 Privacy in today's Online Social Networks

There is a significant work in the literature on privacy issues in today's online social networks. The service providers are always driven by their business goals and fail to recognize to a very large extent, the critical privacy needs of the users and few of the

design choices they made prove that privacy preservation is not their top priority. The Google's Buzz [10] service and Facebook's Beacon feature [11] were classical examples for the same. Privacy breaches are commonplace in other social networks as well [23]. Some social network interfaces offer a default setting as public sharing of a data object and the users explicitly have to change the setting to limit the visibility only to his friends. The work in [84] highlights the disparity in the desired privacy settings on OSNs and the actual settings provided. It also quantifies the process of managing privacy. In [87] authors show that configuring privacy settings in the online social networks is a daunting task. Moreover, often, the privacy policy of the providers is very cumbersome to be read even for legal experts- for instance, Facebook privacy policy is longer than the US constitution [32]. The possibility for involuntary personal information leakage in current social networks is highlighted in [80], e.g., by means of certain OSN features like annotating or tagging user photos, and its effects are demonstrated in [79]. Even though very few OSN users (only 6% [62]) trust strangers with their personal information, operators allow strangers to access a user's profile; e.g., Facebook allows any application developer to access a user's profile [57]. In [79], the authors demonstrate how personally identifiable information- which can be used to distinguish or trace an individual's identity either alone or when combined with other information sources, may leak via the online social networks. This may lead to identity theft and serious sophisticated spamming and phishing based on social engineering. In a famous security- breach on the RSA Security, the hackers employed such a phishing attack to steal passwords [12]. It is well-known that a number of advertising companies install tracking cookies on individuals' browsers when they visit a number of popular news sites. Some tools like Collusion [13] can trace through cookies, as users continue surfing the web and present the observed patterns in installed cookies. The tool reveals that large number of different popular websites are in fact linked to the same few data collectors, who are building up their knowledge of the users' Internet browsing preferences and habits. As the work in [79] shows that all this knowledge can be precisely linked to specific individuals from OSNs. The work in [92] and [132] demonstrates how a user's social profile attributes, even private attributes, can be inferred from the profile data of his friends.

The authors in [80] summarize the number of OSNs (out of 12 considered in their observation) exhibiting certain degree of availability (exposure) for each type of personally identifiable information, which is shown in the Table 2.1. The first column indicates the number of OSNs, where the information is available to all users of the OSN and the user cannot restrict the access to it. This may also be available to non-users of the OSN, and thus the data is *world-readable*. The second column shows the number of OSNs, where it is available to all users in the OSN via the default privacy settings, but the user can restrict the access via these settings. The third column shows a count of OSNs, where users can fill out in their profile, but by default the value is not shown to everyone. The fourth column shows the count of OSNs, where the information is not part of a user's profile and thus the information is not available unless the user adds it on his page.

Table 2.1: Availability of personally identifiable information in a total of 12 OSNs [80]

Personal information	Always available	Available by default	Unavailable by default	Always unavailable
Personal photo	9	2	1	0
Location	5	7	0	0
Gender	4	6	0	2
Name	5	6	1	0
Friends	1	10	1	0
Activities	2	8	0	2
Date of birth	0	4	7	1
Employer	0	6	1	5
Email	0	0	12	0
Phone number	0	0	6	6
Street address	0	0	4	8

There are a number of cases, where significant privacy breaches by OSNs required regulators' interventions [14], [15], [25]. A privacy breach happens every time a user's data is accessed by a 3rd party without complete awareness of the user himself. On several occasions, OSN data was directly available to a number of government agencies [16]. In addition, deletion of certain data by users may not actually get deleted from the OSN storage leading to further privacy concerns. Such issues are not typically the top priority for OSN designers and usually get addressed as an afterthought [17]. Certain new features on OSNs may lead to new type of privacy breaches which users may not be adequately competent enough to understand which are not foreseen by them during the social network sign up process [27]. Even a friend on your social network can share your entire profile with the government [18].

2.5.3 Towards privacy-aware Online Social Networks

Realizing a privacy preserving OSN keeping the functionality the conventional OSNs offer is a huge research challenge and adequately addressed by the research community. These efforts can be categorized into:

1. *Systems that complement the existing architectures:* These proposals allow the users to use the same existing infrastructures (e.g., Facebook.com) yet, with privacy preserving content sharing. They employ either encryption mechanisms or mechanisms that replace the content with fake one as described below.

2. *Systems based on new architectures:* These proposals build privacy preserving infrastructures that completely replace the conventional ones- for example, decentralized OSNs employing access-controlled P2P systems.

Both of these categories are elaborated in the following.

2.5.3.1 Systems complementing the existing architectures

The *Lockr* system [123] improves the privacy of centralized and decentralized content sharing systems. It allows users to control their own social information by decoupling the social networking information from other OSN functionality using social attestations which act like capabilities. However, these social attestations are used only for authentication and authorization is enforced using separate authorization policies. Persona [41] uses attribute-based encryption to realize privacy-preserving OSNs. The attributes a user has (e.g., friend, family member, colleague) determine what data he can access. The private data is encrypted using logical expressions of attributes such as *friend AND colleague*. Each possible combination of all attributes defines a group of members. The advantage of this approach is that each set of data is encrypted once using proper combinations of attributes for a group of members under these attributes. However, complexity lies in defining the attributes for each member and distributing the corresponding keys. The drawback is the computational cost in attribute-based encryption (a public key based scheme that is 100-1000 times slower than RSA) and the cumbersome membership revocation mechanism which involves rekeying all the non-revoked members in the same group. The flyByNight approach [85] encrypts a user's content and hosts it on the OSN using a client-side Java Script based Facebook application. The application uses the Facebook interfaces to perform the key management necessary for the encryption. However, such approaches are vulnerable to active attacks by the service provider itself (i.e., Facebook in this context) as observed by authors in [70]. The NOYB approach [70] adopts a novel approach for preserving content privacy. They observe that if users address their privacy issues themselves by hosting encrypted content on OSNs, they could be expelled from the OSN by the OSN operator. Hence, they propose to replace users profile content items with "fake" items randomly picked from a dictionary. NOYB encrypts the index of the user's item in this dictionary and uses the ciphered index to pick the substitute. The authors in [105] acknowledge increasing awareness of privacy issues in the OSN users and propose a market mechanism for exchanging personal data for monetary gains. In [121], a privacy-preserving scheme for data sharing in OSNs is proposed with efficient revocation of access rights. This framework prevents a contact accessing the private data once the contact is unfriended in the OSN. It also offers efficient search over encrypted content in spite of changes in a group membership. It employs broadcast encryption [59] for efficient access rights revocation, where role-based access control model was used. Broadcast encryption is widely used in the secure distribution of copyrighted media (e.g., TV programs, DVD). Broadcast encryption allows a transmitter to send encrypted data to a set of users such that only an authorized subset of users can decrypt the data. Each member of the OSN is assigned a single role. The owner of a profile has control over access to his/her profile data. He/she would act as a group manager who classifies his friends according to their roles (e.g., family, colleagues) and grants them the corresponding memberships. Each role defines a subgroup, the members of which are restricted to certain data categories. A data category is created

by the data owner describing the set of data files that can be accessed as a whole by one or more subgroups.

2.5.3.2 Systems based on new architectures

Despite the ability to work with existing infrastructures, the above class of solutions, by design, need support of the OSN providers which is quite unrealistic given that encrypting the social profile content conflicts directly with their very business motives. In addition, such approaches have minimal flexibility in exercising various degrees of privacy preservation which still largely dictated by the OSN provider policies. Moreover, those solutions still need certain additional infrastructure in the form of cryptographic key management and the client side tools to perform the encryption and decryption workload. Hence, recently, the issue of using decentralized infrastructures for organizing OSNs in a privacy-preserving manner, was addressed by the research community [45, 50, 88, 114]. In [88], the authors perform an experimental evaluation of hosting OSN content from homes as a possible decentralized OSN with the motive of enabling the users to have full control on their content. They make use of social network workloads from YouTube and Flickr and the data of characteristics of home network infrastructures. In addition to addressing privacy concerns of the OSN users, such an architecture allows users to share content without any constraints for example, on the type of the content- images, videos. PeerSon [45] adopts encryption mechanisms for content storage and access control enforcement. It uses a two-tier architecture in which the first tier is a DHT, which is used as a common storage by all participants. The second tier consists of peers and contains the user data. The DHT stores the meta-data required to find users. Peers connect each other directly, exchange the content, and then disconnect. The work in [50] addresses privacy in OSNs by storing profile content in a P2P storage infrastructure. Each user in the OSN defines his own view ("matryoshka") of the system. In this view, nodes are organized in concentric rings, having nodes on each ring trusted by the nodes on its immediate inner ring with the user node being the center of all rings. The user's profile data is stored encrypted at the innermost ring, which is accessed by other users through multi-hop anonymous communication across this set of concentric rings. In the DHT, an entry for a user with the list of nodes in the outermost ring is added. Thus, [50] achieves both content privacy (using encryption) and anonymity of searcher and hosting nodes, yet limited content discovery and profile availability, as opposed to our approach. DECENT [119] proposes a DHT based storage for OSNs with a special focus on security and privacy using encryption mechanisms. They aim to protect the confidentiality, integrity of the user content- using cryptographic techniques, its availability- using replication techniques, and the privacy of user relationships. DECENT supports flexible access control policies, fast revocation, and ensures that neither the user's profile data nor his relationships are visible to unauthorized users.

Vis-à-Vis, a decentralized OSN, is proposed in [114], where, a user's profile content is stored on his own machine called as virtual individual server (VIS). VISs self-organize

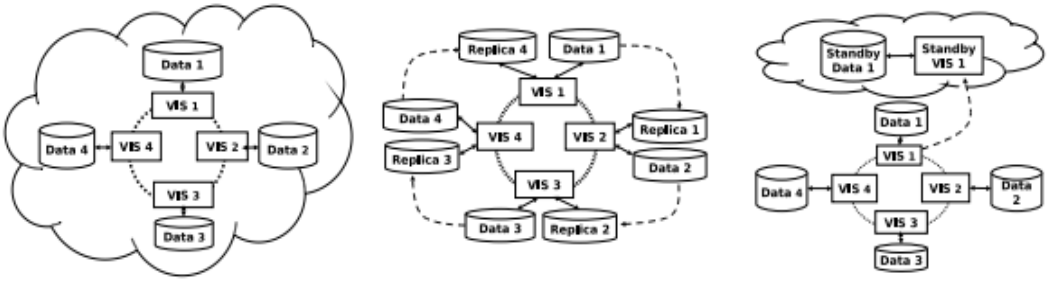


Figure 2.3: The Vis-à-Vis [114] system with 3 different storage environments: cloud, P2P storage of desktops, and a hybrid storage.

into P2P overlays, one overlay per social group which has access to content stored on a VIS. Three different storage environments are considered as shown in Figure 2.3: cloud alone, P2P storage on top of desktops, and a hybrid storage. Their availability, cost, and privacy trade-offs were studied. In the desktop-only storage model, a *socially-informed replication scheme* was proposed, where a user replicates his content to his friend nodes and delegates access control to them. However, normally, a user trusts only a fraction of his friends to the extent of delegating access control enforcement, as considered in our *My3* approach. *My3* also models online times of individual users in the OSN to design a highly available decentralized OSN. It also exploits trust based access controlled P2P system discussed in Chapter 3.

Tribler [93] is a P2P file sharing application which exploits friendship relationships, tastes and preferences of users to increase the performance of file sharing. However, in Tribler, users host their own profile and therefore profile placement for high availability and low access or consistency cost are not considered. Each peer maintains 4 caches called megacaches- friends list, super/peer cache, file meta-data cache, preference cache (of preference lists of other peers). These caches are maintained up-to-date which results in an overhead traffic. All the meta-data of files is replicated among all peers, thus moving content discovery from network-based keyword search to local meta-data browsing. Content discovery is done using taste-buddies: connect people with similar tastes and replicate file meta-data on all peers. A buddycast algorithm is used to exchange preference lists and content of megacaches. Users can disable the functionality of buddycast to protect their privacy. LifeSocial [67] is a P2P-hosted OSN, where users employ public-private key pairs to encrypt profile data that is stored in a distributed way and is indexed in a DHT. Friends can read a user's profile based on a symmetric key that is encrypted with their public keys. However, data privacy and profile availability are not considered.

The Diaspora [19] project aims to build a user-owned decentralized online social network. It consists of independently owned *pods* (or servers) which host user profiles and form the network. However, the Diaspora system needs the pods to be online always.

We believe that the *My3* model for decentralized OSNs, where users can run their clients for a fraction of time compared to always-on availability of Diaspora, is more amenable.

To the best of our knowledge, *My3* is the first system that uniquely identifies the availability of decentralized OSNs as the critical concern for their adaptability and considers user behavior characteristics on OSNs and exploit them in its design.

2.6 Summary

This chapter reviewed the background and related work that lays out the context for the rest of the thesis study. From the sections devoted to individual topics, we can claim that issue of designing access controlled P2P systems powered by search-efficient privacy preserving index is a very attractive topic which has a number of vital applications in practice. Privacy-preserving online social networking is one such application as discussed in last section of the chapter.

Part II

Access Control aware Peer-to-Peer Publishing

Chapter 3

Access Control aware P2P Data Management Systems

The belief that complex systems require armies of designers and programmers is wrong. A system that is not understood in its entirety, or at least to a significant degree of detail by a single individual, should probably not be built.

Niklaus Wirth

3.1 Introduction

The peer-to-peer communication model has introduced a significant paradigm shift in the way users share the resources and communicate with each other on the Internet. Structured P2P systems based on the Distributed Hash Table (DHT) paradigm are increasingly adopted in building massively scalable and highly available data management systems. The academic research community has pursued a wide range of such systems which include- Chord, Pastry, P-Grid, and CAN [37, 102, 106, 120]. DHT based scalable storage infrastructures are embraced beyond the realms of academia also [36]. A number of storage applications such as file systems [52] and backup systems [83] have P2P paradigm inherent in their design. Massively scalable distributed semantic data base stores that store RDF data are built on DHT based structured P2P systems [30, 34, 38, 46, 75].

In spite of increased usage, the lack of handling privacy and security issues, sets clear limits on the adoption of P2P-based systems. For any meaningful applications to be built, security should be the critical design goal for the underlying infrastructure. The foremost important concerns include

- *Securing the communication infrastructure-* This requires that no unintended third parties can eavesdrop and intercept or alter the messages between any two peers and still go undetected. The communication messages exchanged should be visible only to the intended participants.

- *Securing the access to the shared resources-* ensures that only authorized peers have access to shared resources. The access control specification in a system is typically configured using access control lists [68]. A secure system must support trustworthy access control aware search and data accesses. P2P collaborative systems make such task challenging given that they lack any centralized control and are realized on top of untrusted resources or nodes.

The first requirement is typically realized by using the Secured Socket Layer (SSL) [33] based communication channels which make use of public key cryptography. For example, as part of the TEAM project explained at the end of this chapter, the P- Grid [31], a DHT- based P2P system, was enhanced with SSL- based secured communication [94]. This secured communication aspect is not dealt further in this chapter, and we deal mainly with the second requirement and study ways to enhance existing P2P systems to provide means to enforce access control. We refer to, an access control aware P2P data management system, as *ACPeer* throughout the chapter. We study ACPeer systems based on structured P2P networks- the most popular kind of P2P infrastructure, where, for each resource *r-* (example: a file, a data object, etc.), an index item which is in the form of a *(key,value)* pair is stored in the system as explained in Section 2.2. We argue that even these index entries need to be protected from unauthorized accesses in addition to the original resources for which the index is constructed. In the case of DHT based RDF triple stores like the GridVine [38] system, the RDF triples form the *value* field of index which may contain sensitive metadata. In this case, the index hosted in the DHT itself constitutes the resource. Thus our goal is to realize a system, where the owner of a resource can control the access to both the resource and the index. We discuss and compare various approaches and discuss the feasibility of each method in practical realization.

In contrary to the centralized storage systems and the unstructured P2P systems which do not have any index hosted, there exist more than one point in a structured P2P network to enforce access control i.e., implementing the reference monitor. As to be discussed later, the choice of the location of the reference monitor, greatly affects the overhead due to access control, both in terms of the resource search and publication cost. By carefully choosing the enforcement points and the implementation methodology, we can realize systems easy to administer.

The simplest approach to control the access to resources is to have multiple disjoint overlay networks each hosting resources and having an admission control mechanism to allow only the corresponding authorized peers as its participants. However, this solution has inherent scalability problems. Hence, we introduce later, a few alternatives

involving cryptographic techniques that enforce access control at *query* stage itself or at *reply* stage of a query. Finally we present a hybrid solution involving both, which has several advantageous scalability properties.

The remaining of the chapter is organized as follows. Section 3.2 gives a more precise definition of the access control and motivation behind the problem we are studying. In Section 3.3, we describe and compare the various techniques for realizing access control in structured P2P networks. Section 3.5 discusses related work and finally Section 3.6 concludes the chapter.

3.2 Overview of the Problem

In this section, we present the problem statement along with the necessary background. Then we sketch the modeling of various aspects of the system along with the modeling of the access control policy and highlight our approach.

3.2.1 Access control in structured P2P networks

As explained in Section 2.2.1.2, in a structured P2P network, the resources are found using indexing where the index items, in the form of *(key,value)* pairs, are stored in and retrieved from the network using two system provided primitives namely `put(key, value)` and `get(key)` respectively. The `key` field is a key to uniquely identify a resource item in the network and the `value` field is either the physical location of the data object or data object itself. Users/applications use the system to publish and search resources/data objects (files, RDF triples data, for example). Figure. 3.1 illustrates the

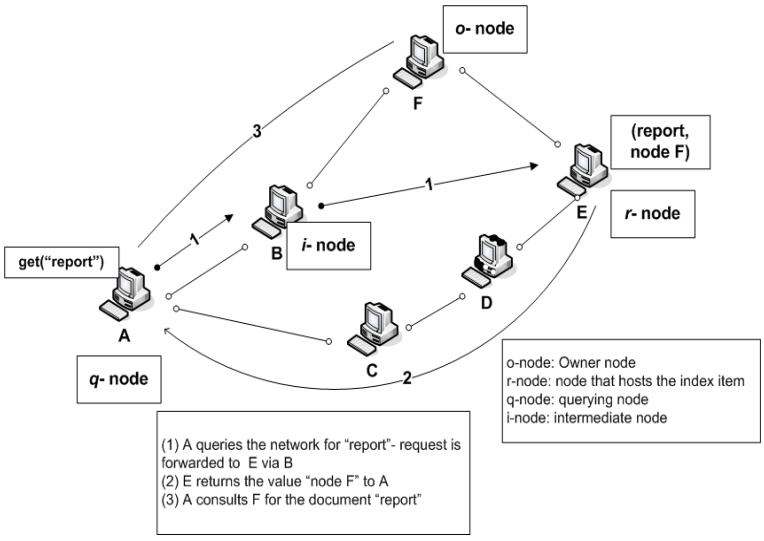


Figure 3.1: A typical message flow in a structured P2P system.

typical work flow of a structured P2P system. A peer F wishes to publish a resource with name *report* in the network. This node is termed as the *owner node (o-node)* in the model. The underlying routing algorithm decides to host the index information

about this resource ($index(report)$, say a pair $(report, F)$) on node E which is termed as the *responsible node (r-node)*. Any peer can query for this resource using the `get` primitive. In the figure, the querying peer (*q-node*) A queries for this item and the query is forwarded to node E via several *intermediate nodes (i-node)*.

In an *ACPeer* system, there exists, for each published resource r,

1. a set of data, to be protected from illegitimate accesses called $\Lambda(r)$ i.e., access controlled data for the resource r, which can include several pieces of data associated with r based on the extent to which the access control is enforced, as shown later in this chapter.

2. an access control policy rule P, which maps r to a set of peers that can access the $\Lambda(r)$, which is more formalized later.

If $\Delta(u)$ is all the data that a user/peer[1] u can access, then an *ACPeer* system must ensure that,

$$\Lambda(r) \subseteq \Delta(u) \text{ if and only if } u \in P(r), \text{ for any resource } r.$$

The *ACPeer* systems can be categorized based on what can constitute $\Lambda(r)$ for a resource r, as explained in the following.

3.2.2 Levels of access control

The set $\Lambda(r)$ for a resource r determines the level of access control the system can provide. Based on various requirements of the applications built on top of *ACPeer* system, we can categorize an *ACPeer* system as one of the following, where for each resource r published,

1. **Level-0** system (L_0) if $\Lambda(r) = \emptyset$. This system does not provide any access control on the resources. Such a system is suitable for applications which need access control-free raw P2P storage. The conventional P2P systems- Chord, Pastry, etc. are examples of L_0 systems.

2. **Level-1** system (L_1) if $\Lambda(r) = \{r\}$. Such a system protects only the resource from illegitimate accesses which is useful, for example, to host document publishing systems where every document can be accessed only by authorized paid members, but the index of the documents is world-readable so that any user can search within the index. Systems like *IEEEXplore*[2] are L_1 systems.

3. **Level-2** system (L_2) if $\Lambda(r) = \{Index(r), r\}$ where $Index(r)$ represents the index of resource r. This index may include in addition to the $(key, value)$ pairs, any sophisticated metadata like descriptors, keywords of the data object. L_2 systems provide the highest control on the published resources. These systems realize access control aware searches [118] on the index.

[1] We use the terms *user*, *peer* and *node* interchangeably throughout the text.
[2] www.ieeexplore.org

In addition to the above classification, one can imagine a system where Λ covers index only partially, where in, access to the *value* field of the $(key, value)$ pair is controlled but the *key* field is not. Such systems can be labeled as **Level-1.5** system ($L_{1.5}$) in this context. Such systems allow any arbitrary user in the system to figure out whether a resource identified by a particular *key* exists in the system or not without revealing anything more about the resource. To be seen later, a L_2 system entails a non-negligible overhead in terms of search and publication cost over a $L_{1.5}$ system.

Our goal is to enhance existing structured P2P networks to realize a L_2 ACPeer system.

3.2.3 Modeling access control policy

In this work, we pursue discretionary access control model (DAC). We assume that each resource has an owner who shares the resource and defines the access control rules. We call the subjects to whom access rights are granted as *principals*. We assume a general modeling of a principal, but the solutions can be adopted to different types of principals such as principals modeled as users and groups of users, principals modeled using relations on a social graph among users (for example, all 2-hop friends on the social graph forming a single principal).

The access rights are expressed in the form of access control lists (ACL). We use the notation $\aleph$ for the set of principals, $\Re^p$ for the set of resources owned by the peer p, $\Lambda(r)$ for the set of data items for a resource r to be access controlled, as defined already. Note that, in an L_2 system, for each resource r, $\Lambda(r) = \{r, Index(r)\}$.

$$ACL^p : \bigcup_{r \in \Re^p} \Lambda(r) \rightarrow 2^{\aleph}$$

For simplicity, we assume that a resource r and its index $Index(r)$ will have the same access control rights in our L_2 system. The solutions do support the other case where each will have its own rights. Thus, the goal of a L_2 system is to enforce the ACLs of all peers in the system. As mentioned already, an agent that executes the ACL on a node is referred as *reference monitor*. Hence, a design for an *ACPeer* system should deal with the issue of implementation of reference monitors for the access control policies of all the publishers in the system.

3.2.4 Assumptions

In order to focus on the original problem of designing an *ACPeer* system, we resort to a simplistic model of certain aspects of a structured P2P system as explained below:

We assume that the resources associated with an index item, if any, are stored with the corresponding $o - nodes$, unless specified otherwise. Thus the *value* field of the index entry $(key, value)$ pair points to the $o - node$. Our solutions are adoptable to other storage models as well, for example, where a resource is stored inside the network on the $r - nodes$.

Since verifying the identity of a principal securely is a critical prerequisite for any access control system, we assume the availability of the same. To be specific, we assume existence of a secured mapping function which maps an access request to one of the principals available in the system, using which the reference monitor can either allow or deny the access request based on the policy statement. However, any P2P system backed up by SSL based communication infrastructure already enjoys such secured identity management in the form of digital certificates. For some of the solution approaches proposed in the next sections, we assume the existence of PKI infrastructure which includes certificate authorities (CAs) who can sign identity and public key certificates of users.

We assume a simplistic threat model where peers try to abuse the system by trying to access the data they are not authorized to access. Otherwise, they cooperate with other aspects of the routing protocol- for instance, we do not assume that peers delete the data they have to host as per the protocol and peers do not abuse with the routing algorithm of the P2P infrastructure. We assume that the communication between the peers is secure. In order to mitigate the effects of deliberately malicious and non-cooperative activities one needs other techniques, which are often complementary to our methods.

3.2.5 Motivation

Before addressing the problem of realizing a L_2 system, where access control on the index data is also exercised, here we provide the motivation behind envisioning such a system. As explained earlier, sophisticated indexing mechanism can be built on top of the basic indexing primitives provided in the form of *(key,value)* pairs. An inverted index of the documents published, is one of such possibilities. P2P information retrieval systems based on distributed inverted indexes are pursued in the literature [86]. The authors in [43] argue that any indexed document can be constructed with ease from the open inverted indexes of the document. Hence, a L_2 like *ACPeer* system would be a first step towards access controlled P2P information retrieval systems. Privacy-conscious storage systems must protect the index of the resources also from the unauthorized peers.

Imagine a P2P information system inside a corporation where managers, employees, and temporarily hired contract staff share the resources over the P2P platform. In an L_0 or L_1 system, even the contractor staff can place queries, per say, appraisal review related docs and infer something like whether the appraisal of an employee is ready or not or is it modified after the employee met the manager, all from the accessible index of the resources [116]. Typically mangers may want only other managers or some employees whom he chooses, to see what he is sharing in the network, despite the actual access to resources is restricted. In a knowledge sharing system, adversaries can query the system with few keywords and infer what kind of knowledge a peer is contributing to the system- say *whether any docs in the system exist about virus xyz?*. A L_0 or L_1 system can easily expose such data through uncontrolled index accesses on the metadata or index. The

authors in [116] demonstrate the threats due to such data leakage in traditional file systems.

In systems like GridVine, a single query translates into multiple queries recursively, when matching semantic mappings or translations are found in the network. In L_2 systems, we can prevent queries from unauthorized peers from going deep into the network recursively.

More importantly, in some systems, there may not be any physical resource associated with the index to enforce access control at the end nodes (i.e., $o - node$s in current context). In this case $r = \emptyset$ and $\Lambda(r) = Index(r)$. The resource itself would be stored in the network e.g., in the form of an RDF triple. The TEAM system [34] is an example of such a system where peers share knowledge in the form of instances of OWL ontologies comprising of RDF triples. In such case, reference monitors must be realized inside the network as such. L_0 or L_1 systems do not control accesses in such cases. Such systems need L_2 *ACPeer* systems.

In an L_2 system, a query from an ineligible peer is stopped as early as possible from a typical successful query-reply cycle which saves on the communication costs. Such a peer does not get any useful reply packets for a query term, it can not proceed further on the transaction flow. In case of very popular resources, usually the load on the serving peer is high. If the queue size on such peer grows beyond an acceptable limit, all future requests may be denied which might include requests from eligible peers. So an L_2 system access control enforcement tries to fill the queue with only authorized peers.

An adversary may query the system for particular keywords and launch Denial-of-Service (DoS) attacks on the serving peers using the information he/she gets in the replies. If index is openly accessible, an adversary can construct the history of who are all searching for the key it is hosting from the queries it receives- and this valuable history must be protected for privacy reasons. In L_2 systems, this issue does not arise, because a random adversary can not know about the index just because it hosts it, unless it has access to it.

3.3 Solution Mechanisms

In this section, we explore the problem of designing an L_2 *ACPeer* system in detail and explore various points in the solution space which we broadly categorize into the following. We provide a detailed comparison of all the mechanisms in later part of the chapter.

1. Independent P2P Networks (IPN)
2. Controlled Queries (CQ)
3. Controlled Replies (CR)
4. Hybrid Solution

Algorithm 3.1: Simplified algorithms for IPN

{The algorithms for a peer p in the system:}
PROCEDURE: PUBLISH()
for all resource $r \in \Re^p$ **do**
  publish $\Lambda(r)$ into a single network of all principals in, *or* networks each with one principal from, $ACL^p(r)$
end for
PROCEDURE: SEARCH(r)
for all network with valid membership **do**
  search for r in the network
end for

The IPN approach addresses the problem by constructing independent networks. The CQ approach realizes the reference monitors in the network using encryption techniques where as the CR approach, in an encryption- free manner. The hybrid solution attempts to inherit properties of both the CQ and CR approaches. These solutions are discussed in detail in the following sub sections.

3.3.1 Independent P2P Networks (IPN)

A simple and trivial way of realizing an L_2 system is by creating a set of independent P2P networks each with its own resources published and the corresponding authorized peers as its members. The IPN approach converts the problem of access control into admission control as described below. One way of realizing the admission control is to control the network membership using digital certificates. The IPN approach is closely analogous to the conventional mailing lists or mailing groups (e.g., Google groups, Yahoo groups). There are two ways to create a new network which depend on which one of the search or publication costs has to be optimized.

Method1- Optimizing the publication cost: In this method, each peer publishes its resources at most once irrespective of the number of principals authorized to access. A network with only peers corresponding to authorized principals is created and resources published into this network are accessible to the $o-node$ and the member principals. In this case, a $q-node$ searching for a resource r, has to search in each of the network it participates. ACL updates are dealt by either admitting new principals into a network or reconstructing networks excluding the revoked principal, in case of revocation.

Method2- Optimizing the search cost: The $o-node$ creates one network per each principal and hence, publishes a resource as many times as the number of authorized principals. But for a $q-node$, the number of networks to search in, reduces drastically. When ACL is updated, a new publication takes place for the newly authorized principals. Revocation of access is not possible in this model as data is directly published to a principal's network, whose access can not be controlled after publishing. This way IPN ensures that index is available only to the authorized peers by modeling the access control problem as an admission control problem. Access control is exercised at the time of joining

the network itself. The problem of admission control in P2P systems is well addressed [78, 110], which is not discussed here.

However, the IPN approach has inherent scalability and efficiency problems as the number of networks needed would explode exponentially. For the first case, the number of networks would be in the order $O(|\Theta|\, 2^{|\aleph|})$, where Θ denotes the set of peers and $\aleph$ the set of principals in the system. In the latter case, it would be in the order of $O(|\Theta|\,|\aleph|)$. The networks and the participants in a network become highly unmanageable as the number of resources and peers increases. However, the advantage with this approach is, one can exercise the finest granular access control, right to the level of per-principal resource access constraints. This is useful for applications that can not tolerate even minimum amount of information leakage to unauthorized peers.

The IPN solution tries to attack the problem of L_2 *ACPeer* system by realizing the reference monitors around *o-nodes* (refer Figure. 3.1) only and hence end up in non-scalable systems. By designing a $q-node$ or $r-node$ centered reference monitors, one can realize scalable and manageable solutions as demonstrated in next sections where the techniques of *controlled queries* and *controlled replies* are introduced.

3.3.2 Controlled Queries (CQ)

The intuition behind the CQ based approach is to embed reference monitors into the index generation process itself by using encryption techniques and then ensuring that only the authorized principals get access to the encryption keys. Such encrypted resource or index items can be hosted safely on any untrusted peer in the network. The index entries are encrypted by $o-nodes$ with different encryption keys for each authorized principal, based on the ACL specification. Only the peers with valid encryption keys can pose queries for the items and decrypt the results received. There is no need of any access control execution on $r-nodes$ in the network, as it is already enforced at query formation itself, thus completing the access control enforcement on index (at $q-node$ itself). Ability to generate a query with valid query terms itself is considered as the necessary authorization required for the access. This way of access control on the index, backed by the control on the resources at the *o-nodes* completes the realization of an L_2 system. Resource identifier encryption schemes were used for censorship-resistant publishing in P2P systems [48].

Based on the encryption keys used, either an $L_{1.5}$ or an L_2 CQ-based *ACPeer* system can be realized. In the following, we first introduce a solution to realize an $L_{1.5}$ system and then deal with an L_2 system using additional encryption keys and increased search cost.

3.3.2.1 $L_{1.5}$ *ACPeer* System: Publishing and Searching

An $L_{1.5}$ system is realized using public key cryptography: each principal in the system should possess a public, private key pair. For an authorized principal, the $\Lambda(r)$ is inserted into the network after encrypting it with the principal's public key. So a peer can search

Algorithm 3.2: Simplified algorithms for CQ

{The algorithms for a peer p in the system:}
PROCEDURE: PUBLISH_Level2()
for all resource $r \in \Re^p$ **do**
  encrypt $index(r)$ with a single secret key for all principals in, *or* with a separate secret key for each principal in $ACL^p(r)$
  publish into the network
end for
PROCEDURE: SEARCH_Level2(r)
for all known/acquired secret key **do**
  query for r encrypted with secret key
  decrypt the result
end for

in all the items published to itself by encrypting the item's identifiers with its public key. In fact, the system allows any peer knowing the principal's public key to search, but the results can be interpreted only by the peers which have the corresponding private key.

Since encryption has to be done for each authorized principal, data items in Λ must be published as many times as the number of such principals specified in the policy for the item. This overhead is in the order of $O(|\aleph|)$. However, existence of several copies of the same piece of data is greatly disadvantageous when resource updates are made. Similarly, a search for interested index entry must be done separately for each principal, the peer has credentials (public and private key pairs) for. Hence, the number of search queries is also in the order of $O(|\aleph|)$.

Potential Dictionary Attacks: The foremost objective of the index encryption algorithm is to preserve the semantics of the search process i.e., to retain the functionality of the *get()* primitive, hence it must enable searchability of the encrypted index through the *key* part of the index entries. Searchability is achieved only when *q-nodes* can reproduce the same cipher text equivalent to the interested Λ item, as was generated by *o-nodes* during publishing. This is possible only by using a deterministic encryption [112] algorithm (such as RSA/NoPadding). A deterministic encryption ensures that a given plain text and encryption key combination always computes to the same cipher text every time the algorithm is applied. The downside of such approach is that the *value* field of index is subject to dictionary attacks. An arbitrary peer, knowing the public key of a principal, can place a search request on the items accessible to only that principal and then construct the complete index entry using a simple dictionary attack. For instance, an arbitrary peer can know where a resource *virus_patch.doc* is stored, by posing the query with the corresponding *key* by encrypting with the public key of the authorized principal (assuming such a principal is known to the arbitrary user). He can later try with a list of known peers and encrypting the peer identifiers with the same public key and comparing which one of them matches to the reply obtained for the search query, thus completing the attack.

However, fortunately, to desist the leakage of the *values* to unintended peers due to dictionary attacks, all the value fields of index entries will be encrypted using a *non-deterministic* cryptographic technique. In a non-deterministic encryption (such as RSA/Padding), a given plain text and encrypted key combination computes to a new cipher text every time the algorithm is applied. However, all such ciphers decrypt to the same plain text. This way it is impossible to figure out which one of known values maps to the encrypted value as every time the algorithm is applied, the cipher text varies. Only an authorized peer which has the respective private key can decrypt the *value* field. Note that if non-deterministic technique is used for the *key* part of index, the searchability is lost because the peers can never reproduce the same cipher originally generated by the publisher.

3.3.2.2 L_2 *ACPeer* System: Publishing and Searching

A CQ-based L_2 system is realized by encrypting the Λ items with a secret key (instead of public key) known only to the publisher and the authorized principals. The secret key generation process which can be done in two ways, dictates the publication overhead and the number of encryption keys required. The publisher generates a single secret key for all the authorized principals for a resource and publish $\Lambda(r)$ after encrypting with this secret key. Alternatively, an encryption key is generated per each principal and the Λ item is published encrypted separately for each authorized principal. The secret keys are distributed out of band or by encrypting with the principal's public key and publishing into the network. The authorized peer queries the system for a search key encrypted with this secret key. Since the secret key is not available to the unauthorized peers, they can not frame valid queries. However, this solution has inherent scalability problems as the number of such secret keys required would be in the order of $O(|\Theta|\, 2^{|\aleph|})$ or $O(|\Theta|\, |\aleph|)$ depending on the two cases discussed above. This is analogous to the IPN approach and thus the scalability issue discussed in IPN approach resurfaces itself in the form of number of encryption keys. In addition, the search overhead also increases significantly compared to the $L_{1.5}$ system, as the number of search queries required to search for an interested item would be in the order of number of secret keys as one peer has to search with each accessible principal and for each potential publisher.

Revocation of access rights: It is clear that the CQ-based approach relies on secured distribution of principals' credentials to the potential publishers in the system. In addition to ACL changes, updates in a principal's encryption keys also result in revocation of access privileges. For example, a principal corresponding to a group of users may update its credentials if the group membership is updated. Revocation of access rights in CQ-based *ACPeer* systems can be realized if and only if additional primitive, namely `remove(key)`, to remove an index entry is provided by the underlying DHT. $\Lambda(r)$ items published with old credentials are removed from the network and republished with new credentials, when a revocation is requested. This primitive must be realized in a secured way allowing only the $o-node$ to issue this operation on the *keys* he/she has published.

Given the fact that the Λ items are stored on arbitrary untrusted peers, implementing a reliable deletion operation is challenging.

CQ-based approach is promising as it does not introduce any changes to the routing protocol of the underlying P2P system. An access-controlled query works like any other query and processing the requests is quite transparent to the protocols, and hence can be applied on top of any existing structured P2P system. CQ approach seems to be more suitable for an $L_{1.5}$ than an L_2 system and by sharing the public keys in a restricted way, one can assume that CQ- $L_{1.5}$ provides more controlled access on the *key* field than an L_1 system. Thus, the proposed $L_{1.5}$ system can be assumed to meet an L_2 system's requirements to a greater extent. Given that one needs the public key of a principal to issue search queries, public keys can be restricted from being distributed indiscriminately in the network. In a typical collaborative working environment, exchanging of public keys is normally done in a controlled manner (somewhat similar to distributing email addresses or personal phone numbers). In such cases, an application which need L_2 semantics can still resort to an $L_{1.5}$ system based on CQ approach to avoid the additional search and publication overhead involved in an L_2 system. Restricted distribution of public keys limit the exposure of the *key*-part of index items to arbitrary peers in the network.

In the next section, we demonstrate the Controlled Replies (CR) approach based L_2 system which pushes the ACL execution into the realms of the $r - nodes$ resulting in a very simplified system free from complicated encryption key management inherent to the CQ-based approach.

3.3.3 Controlled Replies (CR)

In this approach, Λ items are stored in an encryption-free manner, hence, any peer including unauthorized peers, can post queries for interested items in contrary to the CQ-based approach where only the peers that have the encryption keys can pose queries for valid query terms. CR- based L_2 system ensures that queries that originate only from the authorized peers are replied i.e., the $r - nodes$ implement the necessary reference monitor. The related access control list is stored at $r - node$ along with the items. Since access control is exercised at $r - nodes$, we need a trustful mechanism to ensure that a particular $r - node$ does so faithfully. Since traditional structured P2P systems assume all peers to be the equally trustworthy, every peer can be a potential candidate for storing index of the resources published by other peers. Such *anybody-can-host-anybody's-index* paradigm is no longer suitable for realizing faithful reference monitors. To this end, we envision a *constrained indexing* technique, where $o - nodes$ control where the Λ items must be stored. In the following sub section, we introduce techniques based on this philosophy.

3.3.3.1 CR with Trusted Sub Overlays (CR-TSO)

The intuition is that $o-nodes$ can choose a set of peers based on the social relationships and constrain the items to be hosted on these nodes only. This way, trustworthy communities of peers are formed and members of the community decide together to host on each other, the Λ items and implement the necessary reference monitors. It should be noted that these are the groups of publishers i.e., $o-nodes$ and should not be confused with a principal modeled as a group of $q-nodes$.

In this approach, each such community forms a suboverlay on top of the main overlay which is termed as *Trusted Sub Overlay (TSO)*. Each TSO has its own identifier space (equivalent to that of the main overlay) for the published resources by the TSO members. A peer in this model plays two roles: one as a member of the main overlay, the second as a member of one or more TSOs and hence it is responsible for its identifier space as part of the main overlay and that of the affiliated TSOs. A peer publishes any access controlled items into one of the TSOs it is member in, and any open items into the main overlay. A single peer can be member of multiple TSOs. Figure. 3.2 demonstrates the concept of TSO where two TSOs, namely, TSO1 and TSO2, are formed by 3 and 4 peers, respectively. Peer p_9 is a member of both the TSOs. Peers p_1, p_9, and p_3 are part of TSO_1 and another overlay TSO_2 has p_4, p_6, p_8, and p_9 as its members. Node p_1 will publish its data items that need controlled accesses into TSO_1 and any open data items into the large overlay which includes all the peers from p_1 to p_{11}.

Any resource that is published inside a TSO must be identified globally by both TSO identifier and an identifier local to the TSO. We propose to augment the traditional flat identifier space used in conventional P2P systems with a two dimensional identifier space to identify resources hosted inside TSOs- one dimension for identifying the TSOs and the other dimension for the resources with in a TSO. Hence, such resources are identified with a `(TSO_ID,resource_ID)` pair. Resources that can be publicly accessed by every peer are published into the main overlay, which are identified with flat identifiers. One form of constrained indexing where the resource indexes are localized based on the domain names is presented in [71].

In this model, a set of peers together decide to form a trusted sub-overlay, at an arbitrary point of time. A natural choice for such a trusted group of users is all members of a research unit in a huge research organization or a subset of members of a project. There exist several possibilities regarding how the peers meet in the first place to decide on the sub-overlay formation. It can be done in a centralized fashion: a single peer (TSO administrator) can decide to start a TSO and invite other trusted members into this TSO, or in a decentralized fashion: where peers in the community use some admission control protocol to extend the TSO. An automatic TSO formation engine can be built which exploits a social network graph of the peers which forms the TSOs automatically based on some configuration settings, for example, all peers directly connected to a peer can be part of all TSOs created by that peer. Any peer that has the characteristics desirable for a trusted reference monitor can be a potential candidate to be included in a

TSO. It is assumed that a member peer behaves appropriately and cooperates in various aspects of the access control enforcement including but not limited to, responding to the policy updates from the owner nodes, executing access control policy while replying to the incoming query requests, and not compromising the hosted index by sharing with unauthorized peers through out-of-band communication channels. Thus, a TSO can begin with one or more peers and extend as new peers join by sharing the same TSO identifier. Thus the new peers share query load and contribute to the storage of the TSO. New peers can join a TSO in two ways: by *invitation-only* mechanism where a member peer invites a non-member peer to join the TSO, by *self-join-request* mechanism where the new peer itself initiates a *TSO-join* request.

An interesting issue that arises in the CR-TSO approach is, the routing and management of the sub-overlays: whether the same overlay topology used for the main overlay should be used for the sub-overlays or a different one tailored to a TSO's needs should be used. A TSO can use a broadcast topology given that it has a few nodes, while the main overlay continues to be a structured overlay. Each TSO can have its own topology and overlay management mechanism independent of all other TSOs in the system. This model can be interpreted as the main overlay forming a huge substrate into which several independent sub-overlay substrates are plugged. This way, each TSO can be plugged into the big overlay any time so as to make the resources available to other non-member peers selectively, and can be similarly unplugged from the overlay. To be explained shortly, a TSO is plugged into the overlay by publishing some insensitive data about the TSO which enables other non-member peers to discover the TSO and query for the resources published in the TSO.

The *Member list* of a TSO is cached locally on every member peer. Given the smaller size of a typical TSO, updating caches on all members when a member joins or leaves a TSO, can be done using broadcasting.

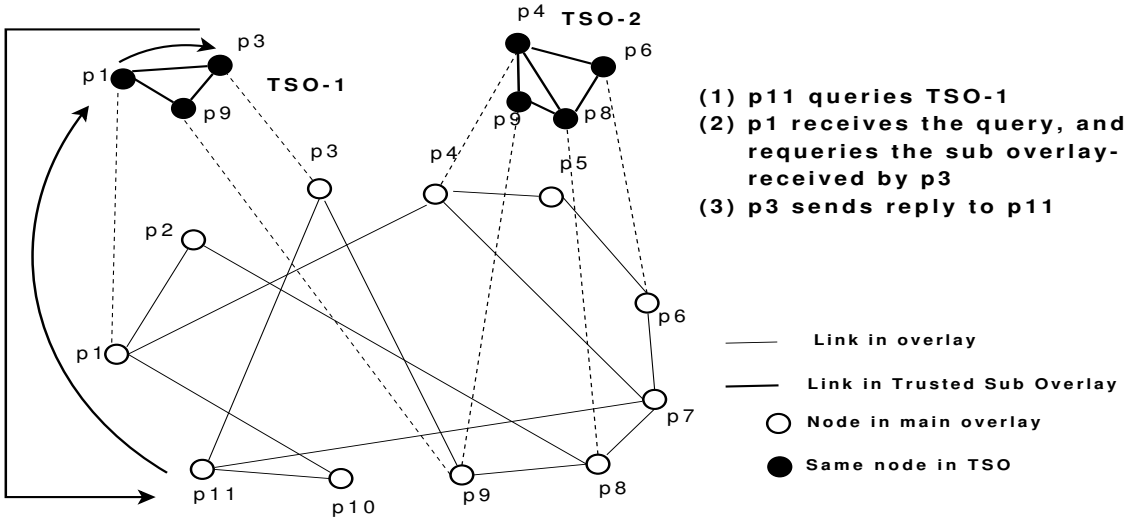


Figure 3.2: Illustration of Trusted SubOverlays.

Publishing: An Λ item is published into a TSO using the TSO-specific primitives for publishing. A publish request message is routed through only the members of TSO. Once published, an Λ item is visible to potentially every member of the TSO. It should be noted that such member nodes may not be authorized to access the resources as per the ACL. However, we believe that this *leakage*, which is measured in terms of number of peers to whom the items are leaked, is tolerable given that trustworthy peers are chosen into a TSO by design. This leakage for a TSO with identifier t w.r.t a member peer m assuming all resources in $\Re^m$ are hosted inside a single TSO, can be quantified as

$$L^m(t) = \sum_{\forall r \in \Re^m} [|MemList(t) - ACL^m(r)| - 1]$$

where $MemList(t)$ is the Member list of the TSO[3]. Thus, the total amount of *leakage* in a TSO, $L(t)$, is leakages of all member peers summed.

$$L(t) = \sum_{\forall m \in MemList(t)} L^m(t)$$

It should be noted that $L(t) = 0$ if and only if either $|MemList(t)| = 1$ or every member of $MemList(t)$ can access every resource published by each of the other members. For very sensitive Λ items, which can not tolerate any leakage, the TSO size can be kept to be only 1 where the items are stored on the owner peer itself. Alternatively, TSO members can be chosen such that the leakage always amounts to zero. The latter way of the TSO formation is rather very difficult as it is not always possible to find such a set of peers which have access to every resource published by each other. Hence, it is always the case that there exists some leakage in a TSO. However, we believe that, since a TSO has typically a fewer number of nodes because TSOs are modeled on trusted social relationships which are limited in number, the amount of leakage is always within tolerable limits.

It should be noted that a single peer can be part of multiple TSOs, each tailored (w.r.t the membership and topology) to the kind of items hosted inside the TSO. A TSO that offers minimum leakage, is chosen for publishing. Once a TSO is selected, either all or some of the resources are published into the TSO. Members of a TSO must publish certain metadata about the TSO into the main overlay so as to enable other non member peers to discover and contact the TSO members. This is done by publishing a list of *entry points* of a TSO into the main overlay. An entry point is any member node of the TSO which acts as a contact point for the TSO. The entry points of a TSO are used in the search process as explained later. Entry point list is a collection of $(TSO\_ID, node\_ID)$ pairs. An arbitrary peer in the overlay must first be able to contact some member of a TSO so as to search for an interested data in the TSO. Entry points serve this purpose. The number of entry points in a TSO need not necessarily be equal to number of member nodes of the TSO as some nodes may want to hide their TSO

[3]'-1' is added because $M(t)$ includes the member m itself which is not present in ACL.

membership information. The authenticity of such an entry point list is of paramount importance for the system to function. One way of realizing it is by using certificate authorities of PKI infrastructure to issue entry point list certificates to all legitimate members of a TSO.

Searching: The best advantage of a CR-TSO is to allow a peer which is not a member of a TSO to query for the resources inside the TSO and access them if authorized. In a CR-TSO, two types of searches exist. An arbitrary peer can search for the resources published into the main overlay using the *get*() primitive as discussed earlier. However, the search for access controlled data that resides inside TSOs is done in two phases. First at least one entry point of a TSO should be retrieved from the main overlay. This is done by querying the main overlay with the TSO identifier. Then the original query should be passed to this entry point. One entry point is chosen randomly in case there are more than one available. Once the query enters into the TSO overlay, that sub overlay's routing mechanism dictates where this query should be forwarded from that entry point. Once the peer responsible for that identifier space sees the query, it invokes the reference monitor which replies to the query. Thus, this simple access control aware search mechanism meets the access control requirements of the L_2 system we sketched in the beginning of the chapter.

However, the important issue arising is, how the $q-nodes$ know about in which TSO to search for when many TSOs exist in the system? It is clear that with the above simple search mechanism, each peer has to search in each TSO separately, thus the number of search queries is in the order of the number of TSOs in the system. This increased query traffic overhead is likely to be limited when the number of TSOs is bounded. However, if the querying peer knows about the TSO it has to search for, this overhead is limited to two queries- first to retrieve the entry points and then the actual query to this entry point. However, as an improvement, we envisage sophisticated techniques where the TSOs publish *data descriptors* which capture general broad description of the data hosted inside a TSO without revealing much about the actual data or index, thus not violating the requirements of an L_2 system. For example, a TSO that hosts a software project related data, can have all the keywords describing the project in the data descriptor. The issue of improved search in the CR-TSO approach, is discussed in detail in later part of the thesis in Chapter 4.

However, for the index which does not require L_2 access control, (key, TSO_ID) pairs can be published into the main overlay so that all the TSOs where a key is available can be retrieved with a single query. Hence, peers will not end up in failed searches in TSOs that do not host a particular key. This improvement is analogous to k- *anonymous* [43] solutions where one can attribute a *key* to a set of k peers (if the size of the TSO is k) but not uniquely to a single peer in this set. At the same time, it provides $L_{1.5}$ semantics for *ACPeer* system.

Mitigating updates in ACL and TSO membership: One very promising advantage of the CR-TSO approach is the ease to handle the updates in ACL. Both addition of new

Algorithm 3.3: Simplified algorithms for CR-TSO

1: {The algorithms for a peer p in the system.}
2: *PROCEDURE: PUBLISH()*
3: {The algorithm assumes that p selects a single TSO for hosting all of its resources.}
4: choose a TSO t out of all known TSOs such that $L(t)$ is minimum
5: **for all** resource $r \in \Re^p$ **do**
6: publish $\Delta(r)$ including $ACL^p(r)$, into t using t's primitives for publishing
7: **end for**
8: [Optionally, join *entry point list* of the TSO]
9: *PROCEDURE: SEARCH(r)*
10: **for all** TSO selected to search in **do**
11: retrieve *entry point list* of the TSO
12: send query message to one entry point for r
13: **end for**

rules and revocation of rights are simple and dealt the same way using a new primitive `put_policy(key, ACL(key))` which is routed to the responsible node which replaces the old ACL rule for the resources, with the new one. The peers cache the identity of the owners with the resources and check whether the policy updates are originated by the owner or not.

Regarding updates made to a TSO's membership, when a new node joins a TSO, it shares the usual query, storage loads as dictated by the overlay management mechanism used in the TSO. A node is excluded from a TSO due to either the member is no longer interested to execute the ACL of the TSO members or the member is found to be breaching the faithful execution of the ACL as found out by out-of-band mechanisms. In any case, the TSO must be reconstructed excluding the member. The excluded member then can pose queries for the data hosted in the TSO like any other non-member peer in the system.

In general, the TSOs are orders of magnitude smaller in size compared to the main overlay. This restricts the replication algorithms from having large number of peers and storage at their disposal for robust replication. In addition, the TSOs need overlay management, which is an overhead. In the following sub section, we propose a light weight approach to the CR-TSO case, which does not involve constructing any sub-overlays.

When looked closely, the CR-TSO approach can be interpreted as analogous to the IPN approach, but it promises several improvements which are not feasible in the IPN approach. Both the approaches share the same philosophy of modeling access control problem as an admission control problem. In the IPN case, we control the members of the whole network where as in CR, the members of a community are controlled. A peer which is not member of a TSO can also issue a query for the items hosted in the TSO. In the case of IPN, a peer must be part of the network and hence, should participate

in all the roles of the network. Consider a scenario where a peer has access to only a single resource. In the case of IPN, a network must be realized for this case. In the case of CR, the peer need not be part of the TSO and still can be allowed to access this single resource alone. It clearly demonstrates that CR-based approach promises increased easiness in managing the access control framework. In the case of revocation, the entire P2P network need to be built in the case of IPN. In the case of CR, only the policy is updated on the peers in the TSO. CR-TSO handles policy updates in a flexible way without changing the overlay state.
Enhancements: However, the CR-TSO approach lacks on some fronts. Since the published data is localized inside a TSO only, which has normally a fewer number of nodes, such data does not match the availability guarantees possible in the huge public overlay due to high number of peers. Higher availability comes with replication. Thus, a smaller sized TSO can not match the overlay w.r.t the level of geographical diversity and the number of peers it provides, which are desirable characteristics for any high available replication technique. Further, management of sub-overlay may add further control overhead. Overlay management overhead includes the cost of maintaining consistent routing table entries, cost of addition or deletion of a node to/from the overlay.

3.3.3.2 CR with Trusted Proxy Set (CR-TPS)

CR-TPS is a light weight approach to the CR based L_2 system. It nullifies the overhead involved in a TSO sub-overlay management by completely getting rid of the sub-overlay, where the members of a TSO, instead of forming a sub-overlay as such, will assume the role of a set of trusted proxies with each member of the set acting as a proxy for the queries targeted at the TSO. Thus each TPS is equivalent to a TSO w.r.t the set of member nodes and the roles. Any potential member of a TSO can also be a potential member for the corresponding TPS. The same semantics of entry points apply to the case of TPS also where a subset of TPS members announce themselves as the entry points/gateways for the TPS, however, with a slight variation of the role assumed by the member peers that do not announce themselves as entry points, as explained later. The main overlay is used for hosting all the Λ items published through a TPS. Thus, CR-TPS approach inherits from the CR-TSO approach, the same flexible way of exercising access control and from the CQ approach, the wider availability of storage. The concept of TPS and mechanism of searching is illustrated in Figure. 3.3, where each TSO in Figure. 3.2 is modeled as a TPS.
Publishing and Searching: On contrary to the CR-TSO approach, member nodes make use of the storage available on the main overlay, instead of the storage within the sub-overlay, for storing Λ items. However, they are encrypted with secret keys, known only to the members of the TPS, before publishing into the main overlay. In this case, the simplest key management approach is using a single secret key for every resource published by the TPS members. This secret key is defined when the TPS is initially bootstrapped. So any new member who joins the TPS later, simply gets access to this

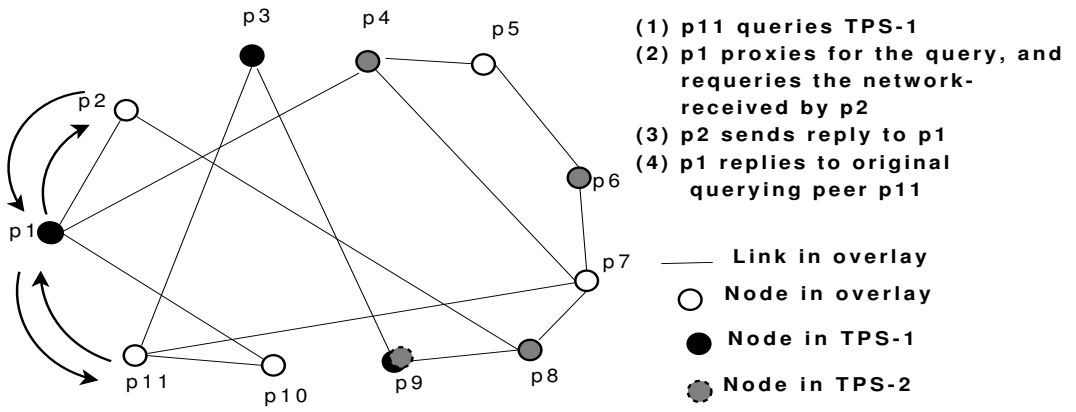


Figure 3.3: Illustration of Trusted Proxy Set.

Algorithm 3.4: Simplified algorithms for CR-TPS

{The algorithms run at a peer p in the system:}
PROCEDURE: PUBLISH()
{The algorithm assumes that p selects a single TPS for hosting all of its resources.}
choose a TPS t such that $L(t)$ is minimum
for all resource $r \in \Re^p$ **do**
  encrypt $\Lambda(r)$ with a corresponding secret key and publish into main overlay
  communicate $ACL^p(r)$ into t
end for
[optionally, join *entry point list* of the TPS]
PROCEDURE: SEARCH(r)
same as the CR-TSO case

key. A searching peer first contacts one of the entry points to the TPS and forwards the query to the node as done in the case of CR-TSO case. The contacted node converts the query into equivalent encrypted query with the corresponding encrypted resource identifier and inserts the query onto the main overlay. Once this node gets the reply, it decrypts the result and sends to the original querying node as illustrated in Figure. 3.3. This way, the members of the TPS act as proxies for the data published by them. In this simple key management technique, the members which did not announce themselves as entry points, do not have any role in the search phase, because such peers will never be contacted by the peers for searches.

However, this simple key management technique leaves the entire Λ items published, vulnerable in case of the secret key leakage to unauthorized peers. Hence, several secret keys can be used with list of mappings of a data item and its corresponding encryption key, stored at every entry point. In this case also, the member nodes which do not contribute to the entry point list will not participate in search phase.

When a peer is excluded from a TPS (for the same reasons discussed for the case of TSO), any new data published after that should be encrypted with new keys which are not shared with the excluded member. However, the excluded member will still be able

to access the data published with old secret keys. To minimize the amount of such data, instead of every member knowing about all the encryption keys used in the TPS, peers share part of the identifier space and the corresponding encryption keys. This approach introduces the overlay among the TPS members, which in turn mimics a TPS as a TSO with an exception on the storage. In this model, members that are not entry points will also participate in the search phase, as the entry points have to contact them for any search queries related to the identifier space shared by those peers.

The CR-TPS approach realizes L_2 systems by building a minimal framework on top of the main overlay. To serve search requests, only one member peer from the entire TPS is enough to be online, which is not possible with the CR-TSO approach. Since items are not stored on the TPS nodes as such, a member can know all of the items published by the TPS members, only by querying the main overlay. On the other hand, in CR-TSO, the items are stored on the nodes and hence exposed to the peers by default. The kind of leakage in the former case (TPS) can be termed as a *hard leakage* where as the latter one (TSO) as a *soft leakage*. Hard leakage is a more desirable property of an *ACPeer* system than a soft leakage.

Another promising characteristic of TPS approach is, the framework can be augmented in such a way multiple peers together can decide on access control requests in stead of a single peer as mentioned above. This can be done using threshold cryptography where a single secret key can be split into multiple parts and shared with multiple peers of a TPS so that more than one share is required to construct the original key.

3.3.4 Hybrid Approach

This approach tries to mix both the CQ and CR approaches together by encrypting the TSO's entry points list and limiting access to only the peers who have the encryption key, thus ensuring a CQ-like access control enforcement on the metadata used for TSO discovery and a CR-like access control for the Λ items hosted inside the corresponding TSO. This way, the system ensures that not every peer in the main overlay can contact a TSO, at the same time ensuring that access to Λ items is controlled.

3.3.5 Qualitative Evaluation

In this section, we evaluate the approaches discussed previously qualitatively and compare them against each other with the following criterion. The summary of the analysis is presented in Table 3.1.

- *Publication cost* measures the number of times a single Λ item must be published to realize its corresponding access control rule specified by the publisher. It can range from 1 (i.e., published only once for all the authorized principals) to as many times as the number of authorized principals.

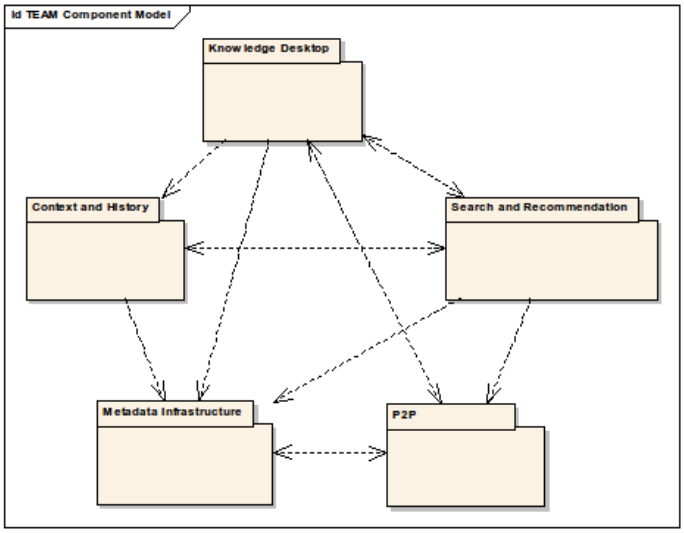


Figure 3.4: Modules of the TEAM system.

- *Search cost* measures the overhead in search in terms of the number of search queries needed to retrieve a specific Λ item. For an L_0 system, it is always 1 and for an L_2 system based on CQ- approach, it will be at most the number of encryption keys the searcher has.

- *ACL updates overhead* highlights the overhead involved in mitigating with ACL updates and revocation.

- *Additional requirements* capture the potential additional requirements imposed by the approach for deployment apart from the requirement of strong identities which is common for all approaches.

- *Storage capacity available* measures the number of peers available at disposal for storing a published Λ item (e.g., due to replication). It captures the level of fault tolerance possible in the system.

3.4 Demonstration of an *ACPeer* System

In the following, we demonstrate an L_2 *ACPeer* system we designed and implemented for the European Union's TEAM (*Tightening knowledge sharing in distributed software communities by applying semantic technologies*) project [34]. It implements the CQ-based approach for exercising access control on the index stored in a DHT which is part of the TEAM storage layer.

The TEAM system envisions a network of users that enables mutual sharing of knowledge and experience for overall improvement of the traditional software development processes. Two forms of knowledge are identified in the scope of the TEAM:

Table 3.1: Qualitative analysis of different *ACPeer* mechanisms

Criterion	Approach				
	IPN	**CQ**	**CR-TSO**	**CR-TPS**	**Hybrid**
Publication cost	1 *or* #authorized principals depending on the criterion (ref. Section 3.3.1)	#authorized principals	1	1	Only once for the Λ items and #authorized principals for the TSO entry point list
Search cost	$O(\lvert\Theta\rvert\, 2^{\lvert\aleph-1\rvert})$ *or* $O(\lvert\Theta\rvert\, \lvert\aleph\rvert)$	#member principals × #publishers	2 ×#TSOs selected (because, once for the entry point list and once for the item search)	3 ×#TSOs selected (since one more query than the CR-TSO case, because of the extra indirection due to proxying)	#cost of CQ + no. of TSOs selected to search in
ACL update overhead	Revocation needs reconstructing the entire private network. Addition of a new rule may require to create a new private network.	Revocation requires a secured *remove(key)* primitive. Addition of a new rule involves either publishing to new principal or sharing a key with it.	Just the policy must be updated with in the TSO. No extra publishing.	Same as CR-TSO.	#cost of CQ + #cost of CR
Additional Requirements	Admission control protocols.	Key management and distribution mechanism.	A light weight admission control with lesser no. of peers.	A light weight admission control and secret key management techniques.	Requirements for CQ and CR approaches.
Leakage to unauthorized peers	Zero	Zero	Zero with appropriate members in a TSO. Non-zero, in normal case.	Same as CR-TSO	Same as CR-TSO.
Available storage capacity	Significantly lesser compared to $\lvert\Theta\rvert$	$O(\lvert\Theta\rvert)$	Significantly lesser compared to $\lvert\Theta\rvert$	$O(\lvert\Theta\rvert)$	Same as CR-TPS.

- **Classical explicit knowledge**, which exists in the form of formal artifacts such as documents or source code. TEAM strives to foster sharing those artifacts that are nowadays buried in private workspaces and not available for interested peers.
- **Implicit knowledge**, that gets seldom formalized. TEAM strives to capture parts of this knowledge automatically. For example, knowledge about a compilation error and subsequent steps taken to resolve it.

The TEAM system represents this knowledge using semantic technologies: OWL and RDF [72]. The Web Ontology Language (OWL) specification defines a language to represent an ontology which is a set of concepts in a specific domain. Each ontology concept in turn consists of a set of Resource Description Framework (RDF) triples. Each RDF triple represents a statement about resources and formally expressed in the form of *subject-predicate-object* expressions.

The granularity of a single knowledge item in TEAM is an instance of an ontology concept (for example, CompilationError) which will be stored as RDF triples in a peer's semantic database that constitutes the TEAM's metadata infrastructure. The system's objective of sharing the knowledge is achieved by publishing a part or all of the metadata stored in the metadata stores of individual users into a network for possible search and retrieval by other users in the network. However, if the content published is not ensured for legitimate accesses by the members of the network, the whole system renders useless as individual user's privacy concerns are not met. In a typical collaborative knowledge sharing environment, a user is willing to give access to his data selectively, for example, only to known people or only to members of a certain project or work location. It is up to the underlying publishing mechanism to realize this requirement and provide user-friendly tools for the users to specify their access control concerns.

We used Jena [90] to realize the metadata store on each peer as part of the TEAM project. Jena is an open-source RDF Store widely used by software communities for storing semantic data. The P2P module, part of the TEAM architecture, is responsible for sharing a part of this local metadata store with other peers in the network. However, the owner peer of this RDF data has access control requirements for this shared data which must be exercised by the P2P infrastructure. Note that this RDF data once published will be hosted inside the DHT. The TEAM project employs P-Grid [37] structured P2P system for this purpose.

3.4.1 Subjects and Credentials

In the TEAM access control model, a *subject* is either an individual or a group of users. A publisher can select one or more of the subjects and configure the system to give access to the published RDF triples only to these subjects. Each TEAM subject is assigned a unique identifier and a public, private key pair as its credentials. Thus a credential can refer to a user credential or a group credential. A user can be part of more than one group and thus can possess multiple group credentials.

To generate unique identifiers in the TEAM P2P environment, we take a simplified approach: a user's email address is used as the unique identifier for a user subject. Similarly, a set of users can form a group and generate a group certificate for themselves. Whoever has the credentials of the group is assumed to be the member of the group and can access all the resources published to the group. The *(email address of the owner of a group, group identifier)* pair uniquely identifies a particular group subject. There is no hierarchy of groups realized in TEAM. There exists a system-defined subject represented by *Entire-TEAM-Network* for publishing content to be accessible to all members of the network.

3.4.1.1 Credential Management

The TEAM system provides tools and necessary services to create, distribute and maintain credentials of various subjects in the network. A user must have an X.509 certificate to participate in the TEAM network from a valid trusted Certificate Authority. The user can use this certificate as his own credential. A user may wish to start a group and invite other members by sharing the group's credential with them. For example, a project manager starts a group in the project name and shares this group's credentials with members of the project. It should be noted that a member of a group can leak the credentials to an arbitrary user and thus making him part of the group, without the knowledge of the group's administrator. Such insider attacks are a cause of concern for every secured system.

When a member is excluded from a group, the administrator will create a new credential for this subject and propagates to other potential publishers. However, the already published data is not re-encrypted with the new encryption key since it involves significant overhead on peers and the network. Thus, the old published data is still accessible to the excluded member which is anyway, a reasonable assumption since the excluded member could have downloaded the data already.

3.4.2 Access Control

We only consider READ access rights; WRITE and UPDATE rights are not considered. The access control policy is specified using an XML schema. The TEAM system enforces the access control policy using cryptographic techniques. RDF triples published into the network will be encrypted while publishing and decrypted after retrieval. A publisher willing to publish RDF triples to a certain subject must first obtain the subject's public key either in the form of a trusted X.509 certificate or directly a public key authenticity of which is verified through out-of-band means. The triples are encrypted with the subject's public key and published into the DHT. While searching, the search key is also encrypted with the subject's public key and a query with the resulting cipher text is posted onto the network. Once the results are available, they will be decrypted with the subject's private key.

3.4.3 Implementation details

This section discusses the system architecture and various implementation details.

3.4.3.1 Architecture

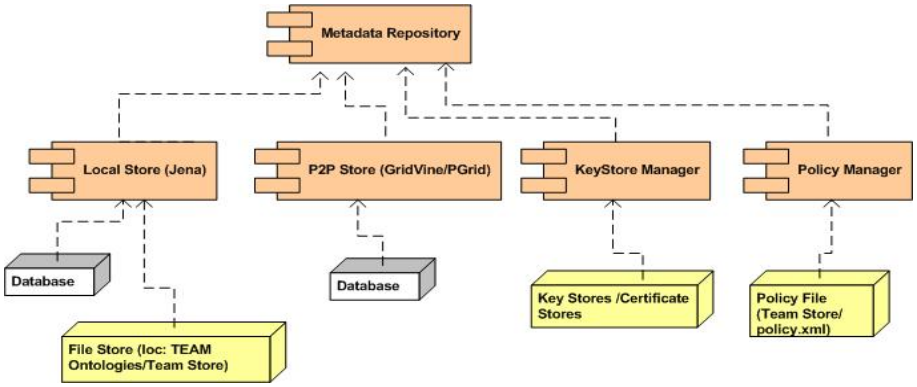


Figure 3.5: The TEAM Semantic Data Repository architecture.

The TEAM storage sub-system (Metadata Infrastructure in Figure 3.4) consists of four components- namely: LocalStore, P2PStore, KeyStore Manager, and Policy Manager as shown in Figure 3.5. The P2PStore component abstracts over the underlying GridVine/P-Grid system and realizes an Access Control aware P2P RDF Store using the services of KeyStore and Policy Managers. Each of these components is explained in detail here.

3.4.3.2 KeyStore Manager

The KeyStore Manager is responsible for providing up-to-date status of various subjects the publisher knows and the corresponding credentials. All the subjects credentials are stored in a keystore with a system-defined name TEAMKeyStore. All the Certificate Authority's credentials are stored in TEAMTrustStore.

The KeyStore Manager parses the TEAMKeyStore and builds an internal up-to-date Subject-KeyPair table. The Manager detects any changes/updates to the store and refreshes the table. The KeyStore Manager provides following services:

1. *getSubjectsForPublish()*: returns a list of subject ids to which the user can publish data. As mentioned earlier, the user needs to know a subject's public key so as to publish to that subject. The returned list also includes Entire-TEAM-Network subject.

2. *getSubjectsForSearch()*: returns a list of subjects for which the user has a private key so that he can search through these subjects. This also include Entire-TEAM-Network subject.

3. *encrypt(data, subjects)*: encrypts the input data with the subject's public key and returns the cipher text.

4. *decrypt(data, subjects)*: decrypts the input cipher text with the subject's private key.

The KeyStore Manager is configured with RSA asymmetric cryptographic algorithm for the above encryption and decryption services.

3.4.3.3 Policy Manager

The Policy Manager provides services to configure the access control policy, manage it, and update it. This XML file is marshalled into Policy Manager's data structures during application start up and unmarshalled back to the XML file for persistence during application shut down. The Policy Manager provides the following services:

1. *getRules()*: returns a list of access control rules present in the existing policy.

2. *updatePolicy(rules)*: appends the set of input rules to the current policy.

3. *validateRule(rule)*: validates the rule against the policy schema.

3.4.3.4 P2P Store

The P2P Store is the key component of the TEAM storage which provides all interfaces for secured publishing and searching. It abstracts the underlying GridVine/P-Grid infrastructure and tries to make the underlying P2P technology as transparent as possible to the other components of the TEAM system. The P2P Store services can be broadly classified as Publish and Search. The process of the user defining an access control rule in TEAM is shown in Figure 3.6.
P2P Publish: Once the TEAM system on a user's machine connects to the P2P network, a background process is initiated by the P2P Store. This process, at regular intervals, checks the policy using the service of the Policy Manager and fetches matching data from the Local Store (refer Figure 3.5), and publishes the data to the subjects. When publishing particular RDF triples, they are encrypted using KeyStore Manager services and the cipher data is published into the network.
P2P Search: This service realizes the following two ways of search in the P2P network. *searchP2PStore(classUri, criterion on properties, subjects)*: This method takes an ontology concept URI as input along with a filtering criterion. In addition, it takes a list of subjects whose credentials are used to encrypt the search queries posed onto the P2P Store. It later decrypts the results with the private key of the subjects before passing the results to the application. The results include all the URIs corresponding to the RDF instances authorized to the mentioned subjects and that match the given criterion. The method *searchP2PStore(instance uri, subjects)* is used to retrieve the RDF instances identified by these URIs.

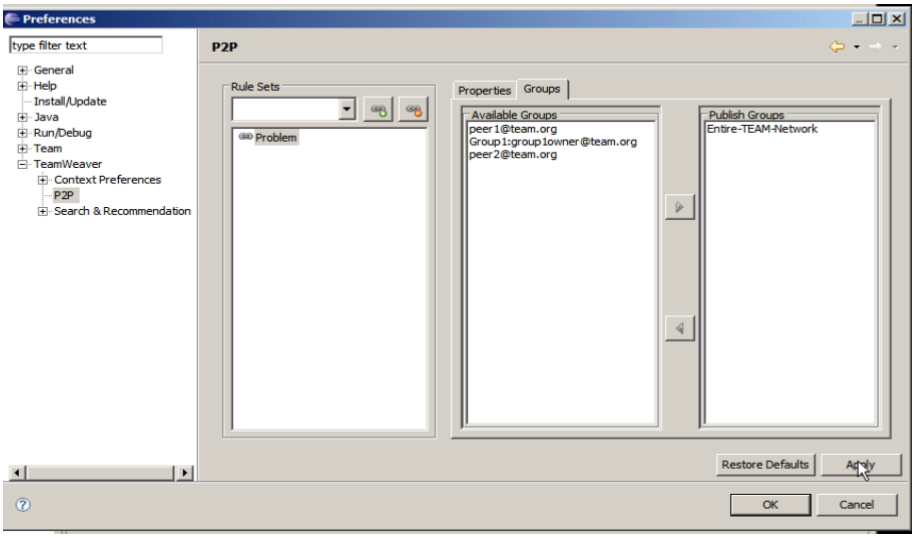


Figure 3.6: A user defining an access control rule in the TEAM client: The RDF resource *Problem* can be configured to be accessible to a subject, per say, *peer1@team.org*. This subject represents a group of TEAM users in this context.

3.5 Related Work

The problem of controlling accesses on shared data is widely studied by the research community which was briefed in Chapter 2 (in Section 2.3). In this section, we discuss the literature which the proposed solutions in this chapter either complement or improve significantly. There is a very little work done in the literature on access control mechanisms for structured P2P systems, whose presence is increasingly expanding into a number of different application domains. The work in [51] provides a policy based access control framework for P2P grid systems. An access control system for collaborative environments involving mobile and P2P systems is addressed in [58]. Access control satisfiability is used in [125] to compute the trustworthiness of acquaintances in a P2P overlay network. Modeling access control in the case of P2P collaborative systems is addressed in [131]. The authors propose a fine granular and attribute based access control framework where each peer assumes a group role and an application role. Then an access control policy which maps various roles and permissions is configured and the underlying framework executes the policy. Our work is based on the lines of PHera [49] which proposes a scalable and fine-grained access control framework for P2P infrastructures. They deal with super peer based P2P overlays where, sub-peers specify their access control policy and the super peer enforces it on their behalf. Any invalid request for a resource will not cross the super peers and reach the peers. Policy statements of individual policies are grouped for scalability and performance reasons. However, this work assumes that all super peers are trustworthy to enforce the access control policy of the sub-peers. In any case, the sub-peers, before processing an access request, can still verify the access control decisions of super peers. As we argue in the chapter, access control aware index accesses are also a critical concern for P2P systems, an issue the

PHera system overlooks completely. In addition, unlike the case of super peer based P2P system, there exist multiple network points to enforce access control in a structured P2P system, as demonstrated in our work.

3.6 Conclusion

In this chapter, we dealt with the problem of access control in structured P2P networks and provided a detailed treatment of the solution space. Various points in the solution space are compared qualitatively. The controlled queries approach was adopted in the TEAM project funded by EU, where an *ACPeer* system was built on top of a P-Grid P2P network. For the ease of reference monitor implementation and the flexibility of handling the updates and revocation of the access rights, the CR-TSO approach must be preferred over the CQ-based approach. However, as acknowledged in this chapter, this approach suffers from search efficiency problems. A searcher has to know a priori which TSO he has to search in. Otherwise, the search request should be broadcasted over all the TSOs. We undertake realizing sophisticated search mechanisms for the CR-TSO case as the next research problem to be addressed and propose a solution in Chapter 4.

Part III

Privacy Preserving Indexing of Access Controlled Data

Chapter 4

PANACEA: Tunable Privacy for Access Controlled Data in P2P Systems

Access to the published data should not enable the attacker to learn anything extra about any target victim compared to no access to the database, even with the presence of any attacker's background knowledge obtained from other sources.

Dalenius- 1977

4.1 Introduction

Peer-to-Peer (P2P) systems are increasingly used in many distributed application domains, e.g., content distribution, file sharing, open storage grids, and video streaming. However, users typically expect to be able to use these systems to share access-controlled and (semi-) private data. Conventional P2P systems should be properly adapted to meet the access-control requirements of such applications. Typical approaches for data access-control in open environments include cryptographic methods [48], Digital Rights Management (DRM) technologies, and trust-based methods [95], which require complicated cryptographic key distribution and management. We consider a simpler, yet effective, approach for data access control in P2P systems: We assume that resources reside at the publisher node itself, to ensure that access control is enforced safely in an untrusted P2P environment. A user directly presents his credentials to the publishing peer of a particular resource after locating the resource in the P2P overlay. The publishing peer replies the query after applying its local access-control policies.

P2P systems typically try to maximize their search efficiency. Structured P2P systems such as Kademlia [89], employ an index implemented as a Distributed Hash Table (DHT) over the P2P overlay. As mentioned earlier in this thesis, such an index typically

consists of index entries of the form $(key, value)$-pairs, where the *key* is the resource identifier (often produced by one-way hash functions, e.g., MD5), while the *value* is the peer identifier, where the resource is stored. Indeed, as shown in [89], such an index significantly reduces the search costs. However, as index entries are hosted on arbitrary and often untrusted nodes, access to the index entries cannot be controlled by the peers that publish their data to the index. Any process of data publishing should not enable an attacker or an unauthorized user to learn more about a resource or its provider compared to a user who is authorized to access [53]. The index published inside a DHT, reveals both the existence/non-existence and the location (i.e., publishing peer in our case) of each queried resource, hence, data privacy is breached. We define the former privacy aspect concerning resource existence/non-existence as *resource privacy*, while we refer to the latter one concerning resource location as *provider privacy*. On the other extreme, unstructured P2P systems such as Gnutella, employ no index. As explained in Section 2.2.1.1, limited-hop flooding is used for locating the queried data which incurs high latency and communication overhead, yet, with no guarantees on the data discovery. However, when access-controlled, unstructured P2P systems can provide the highest data privacy by answering queries only to authorized users. Thus, there is a *trade-off* between search efficiency and data privacy in this context.

In this chapter, we explore this trade-off and propose a PrivAcy preserviNg Access-ControllEd (PANACEA) P2P system that combines high data privacy (both resource and provider privacies) and high search efficiency for authorized users. We quantify privacy offered by PANACEA, employing both probabilistic modeling and information-theoretic approaches. We also analytically study the search efficiency/overhead of PANACEA, as related to structured and unstructured P2P systems. The parameters of PANACEA can be tuned so that the trade-off between privacy and search efficiency is set according to the application needs. Numerically evaluating our analytic results for practical systems and verifying them with simulation experiments, we demonstrate that, with proper values for the parameters of PANACEA, authorized users almost always find the queried resources with a very low search overhead; on the other hand, unauthorized users can deduce the existence of a resource and its provider with a very low probability. Moreover, the communication overhead is high for unauthorized users. Figure 4.1 illustrates the position of PANACEA as related to structured and unstructured P2P access-controlled systems in the three-dimensional space <*provider privacy, resource privacy, search efficiency*>, employing the terminology of [103] described in Section 2.4. On the probabilistic scale, a probability of 0 refers to the case that the adversary cannot deduce the existence/nonexistence of a resource. A structured P2P system provably exposes both the resource existence and provider identifier but offers high search efficiency. On the other hand, an unstructured system guarantees full privacy for both the resource and the provider but with low search efficiency. To the best of our knowledge, PANACEA is the first approach that concurrently addresses resource and provider privacies in access-controlled systems. Note that the specification of authorization pol-

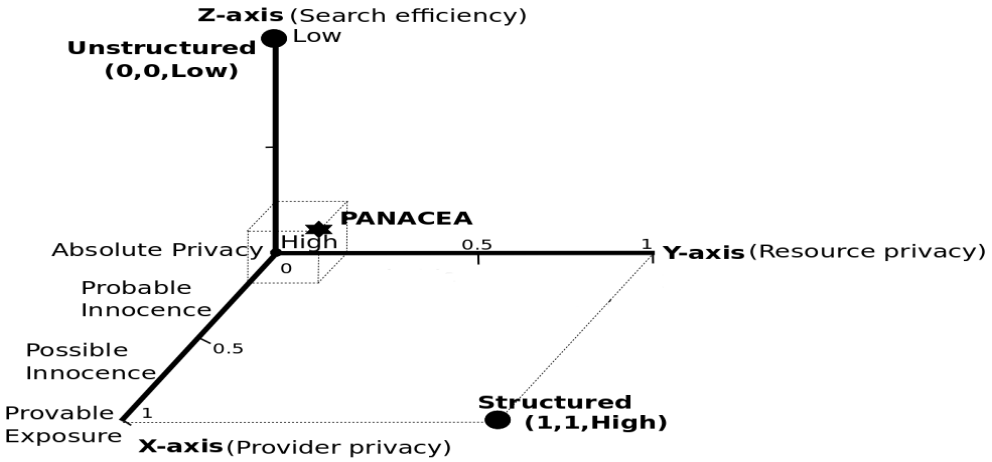


Figure 4.1: Position of various systems on privacy and search efficiency axes.

icy and the user credentials is orthogonal to the scope of the thesis. As a result, the PANACEA mechanism can be employed by providers with different access control techniques, such as role-based access control, discretionary access control or attribute-based access control, all existing in the system simultaneously.

The remainder of the chapter is organized as follows: In Section 4.2, we describe the publishing and searching mechanisms in PANACEA. In Section 4.4, we analytically derive the privacy properties and the search overhead employing a probabilistic approach. We also quantify the resource and provider privacies using an information theoretic approach. In Section 4.5, we verify our analysis and present simulation experiments that demonstrate the effectiveness and the tunability of the system. We briefly summarize the initial experiments we performed on a PANACEA prototype as well. In Section 4.6, we discuss the related work, and finally we conclude in Section 4.7.

4.1.1 Motivation

PANACEA's design is motivated by the fact that resource privacy is essential to any privacy preserving system. It complements the provider privacy as a side-effect and makes the system more privacy efficient. As a result, the maximum possible breach in provider privacy will be limited by the resource privacy achieved by the system. This is the most desirable characteristic of the system for applications publishing resources which have unique properties that make identifying the provider trivial.

The PANACEA system can be used for the following applications: *Privacy preserving web-caching and sharing* application allows communities of users to selectively cache the pages locally and share later using the PANACEA system, which builds the index of the URLs of the web pages cached in the system and the list of providers where the URL is cached. A *privacy preserving user directory service for online social networks* can be built where the social network human readable profile names forming the resources and the profile URLs the locations of the profiles. Another application scenario, where

PANACEA will be relevant, is *user search logs* which reveal all the websites a user accessed in the past. PANACEA promises privacy enabled sharing of such sensitive user logs.

4.2 The PANACEA System

In this section, we present the proposed PANACEA system and explain how the resource and provider privacies are achieved. As already mentioned, resource privacy concerns hiding the *existence* and the *non-existence* of resources i.e, the presence information: an unauthorized user should not be able to determine either of them, regardless of the fact that there can exist multiple instances (henceforth referred to as *copies*) of the same resource owned by different peers. A *copy* of a resource is formalized in Section 4.4.2.1. Our system aims to combine look up efficiency of structured P2P systems with high resource and provider privacies offered by unstructured ones. PANACEA employs a DHT that hosts a resource and provider privacy-preserving (RPP) index. However, as explained later in this section, PANACEA indexes only a subset of the resources into the DHT; this is a necessary characteristic for providing resource privacy. The rest of the resources are located by flooding, similar to the unstructured P2P systems. As a result, PANACEA acts partly as a structured P2P system and partly as an unstructured one.

The proposed indexing mechanism consists of tunable parameters that allow the application designer to choose between strong privacy guarantees and increased search efficiency based on the specific application needs. This tuning of the privacy parameters determines the position of the resulting system in the graph of Figure 4.1, as compared to structured and unstructured P2P systems. We describe the publishing and search mechanisms of PANACEA in Section 4.2.1 and in Section 4.2.2 respectively.

4.2.1 Privacy preserving publishing

PANACEA achieves the resource and provider privacy goals with a novel privacy-aware publishing mechanism, which involves:

1. Probabilistic publishing of resources
2. Resource and Provider Privacy preserving (RPP) index generation
3. Randomized forwarding
4. Insertion into the DHT

The approach is illustrated in Figure 4.2 and is described in the following:

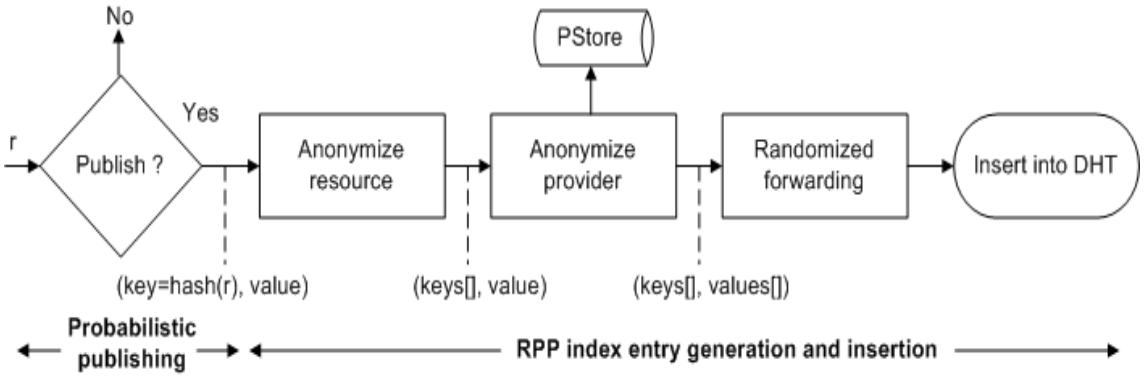


Figure 4.2: Privacy preserving publishing in PANACEA.

4.2.1.1 Probabilistic publishing

Instead of publishing every resource into the DHT, as in structured P2P systems, PANACEA announces a resource with a system-defined probability μ (as shown in Figure 4.2) and creates an RPP-index entry as described later. Therefore, absence of an index entry for a specific resource key in the DHT does not necessarily mean non-existence of the corresponding resource in the system. Due to probabilistic publishing, PANACEA acts as a hybrid semi-structured P2P system. All the resources, which are not announced into the DHT, are discovered using limited-hop flooding with a system-defined time-to-live (TTL).

For some systems, resource's non-existence information may not be sensitive and does not warrant protection from unauthorized users. The value of μ can be set to 1 in such systems.

4.2.1.2 RPP index generation

We employ *k-anonymization* techniques [122] to achieve both the resource and provider privacies for the resources selected to be announced into the DHT in the probabilistic publishing phase. A k-anonymization technique typically anonymizes a data item by hiding it inside a list of k data items so that an adversary cannot identify it. Specifically, instead of having a $(key, value)$ pair as an index entry for a resource, as in structured P2P systems, we propose that the index entry consists of a list of keys and a list of values, i.e., $(key[], value[])$, which is derived by applying *resource* and *provider anonymization* that are subsequently explained. We refer to such an index entry as (m, n)-*index entry*, where m refers to the cardinality of the key-list and n refers to that of the value-list. In this terminology, an index entry of the conventional structured P2P systems can be seen as a $(1, 1)$-index entry.

Resource anonymization: Regarding resource anonymization, once a new resource is chosen to be published, its corresponding $(1, 1)$-index is converted to an $(m, 1)$-index by adding $m - 1$ number of resource keys (that may correspond to *genuine* or *phantom*

resources). Note that human-readable plain text keys (i.e., resource names), which are employed by the users to refer to the resources, are mapped by a hash function to the system key space (i.e., resource identifiers). For a resource namespace R, the equivalent resource key space as K, and the hash function as $H : R \rightarrow K$. The resource namespace R can be domain-specific or span multiple domains and even contain words and their combinations from a dictionary.

When a resource with name r is selected for publishing, there exist two possibilities for anonymizing the resource: *random* and *fixed* anonymization, which are discussed in detail in Section 4.3.

Provider anonymization: After resource anonymization, the resulting $(m, 1)$-index entry is fed to the provider anonymizer module, as depicted in Figure 4.2. The provider list is populated with n number of entries with the providing peer itself being one of them. The other $n - 1$ entries are randomly chosen from the *Provider Store (PStore)* - a local database of provider identifiers. We assume that *PStores* at each peer are initialized with a number of well-known peers and its neighbors in the overlay, and then incrementally expanded over time with stranger provider identifiers contained in the (m, n)-entries traversing through the peer as part of randomized routing explained below.

4.2.1.3 Randomized forwarding

After an (m, n)-index entry is constructed by a publishing peer, it has to be inserted into the P2P system using the DHT *put*() method. However, this index entry must be published *anonymously*, as the next-hop node in the DHT routing could easily deduce that the initiator node is itself the publisher from the (m, n)-entry where it is contained. In order to anonymize the node that initiates the insertion request, we propose that a *randomized forwarding* phase (see Figure 4.2) should precede the DHT *put*() operation. Specifically, each peer that receives the insertion request decides with a system-defined probability λ to *forward* it to a node randomly selected from the n providers (which includes the node itself) in the (m, n)-entry or *initiate* the DHT routing with the $put(m, n)$ method with probability $1 - \lambda$. The technique of randomized forwarding to anonymize the original sender was introduced in Crowds [103] for anonymizing a user's web accesses. A set of cooperating users form a crowd and forward a web access request among themselves before submitting to the web server. The web server can not determine the original initiator of the web request with certainty. However, in [103], the next-hop node was randomly selected from the full set of cooperating nodes before contacting the web server. Clearly, our case is more complicated than the Crowds one, since the (m, n)-entry contains the publisher itself. Hence, by randomly choosing the next hop from the set of providers in the (m, n)-entry, we achieve equal probability for each of them for being the publisher. Note that randomized forwarding precedes DHT routing, and hence it does not demand any modifications to it. The randomized forwarding phase introduces additional communication overhead. This process can be viewed as a geometric distribution with parameter λ. Therefore, if X is the random variable that describes the number of

hops of a *put*() request, then the probability that it travels x hops before it enters DHT routing is given by:

$$P(X = x) \quad = \quad \lambda^x(1 - \lambda)$$

The expected number of hops can be given by the mean of the geometric distribution, i.e., $E(X) = \lambda/1 - \lambda$. We assume that *PStore* (explained in previous section) caches the IP address along with each provider identifier and that the IP address for each provider is stored in the provider list of the (m, n)-index entry. Thus, the relaying of a *put*() message can happen in $O(1)$.

4.2.1.4 Insertion into the DHT

Finally, for the insertion of the (m, n)-entry into the DHT, $put(m, n)$ operation is invoked, which is implemented as follows. Note that the conventional *put*() method inserts only a $(1, 1)$-index entry. The same method can be used to insert a $(1, n)$ entry, as the *value* field is not used in the DHT routing. Hence, we propose to convert the $put(m, n)$ request into m number of $put(1, n)$ requests, using each of the m keys as pivot ones.

Note that, since the keys in an index entry are chosen independently by peers, key collisions are possible. Key collisions also happen when multiple providers of a single resource insert it into the DHT. We propose that the list of providers in the new (m, n)-index entry is simply appended to the list of already existing providers for the collided key.

4.2.2 Searching

When a peer searches for a resource with key r, it executes *get(r)*. If an (m, n)-entry was published previously into the DHT having r as one of its m ids, then the peer returns the provider list of this entry to the searcher. Subsequently, the searcher contacts all these providers. Note that, in general, a user does not know in advance to which providers he is authorized to for the resource r, unless he has contacted them in the past for the same resource. In the latter case, the searcher could select only certain nodes from the provider list to contact. Once an index entry is found, a provider can be reached in $O(1)$ (as in [42]). However, in case of multiple providers for the same resource, an (m, n)-index entry for an existing resource may not contain all the providers of that resource in the system because of probabilistic publishing in PANACEA. In other words, the index entry is not always *complete* because some providers of the resource do not appear in the DHT entry for the resource. As a result, a searcher may not be able to reach the provider where he is authorized through the RPP index. Therefore, even if an (m, n)-index entry is present in the index, the searcher may have to employ limited-hop flooding. However, the probability that a query has to be flooded over the overlay can be very low with proper selection of the publishing probability μ as shown in Section 4.4. We assume that no privacy-conscious provider responds to search queries from unauthorized users in order not to compromise the resource and provider privacies.

4.2.3 System description through an example

Let us assume a provider p_1 decides to publish a resource by name *madonna.jpg*. Let the corresponding *(m,n)-* entry be ($[madonna.jpg, warandpeace.pdf, dangerous.mp3]$, $[p_1, p_2, p_3]$). By definition it is not possible to determine with certainty the actual resource being published and the identity of the actual publisher by accessing this *(m,n)-*entry. All the resources are equally likely to be published and equally likely to be non-existent but picked as part of anonymization. For publishing, this entry is circulated among the nodes p_1, p_2, and p_3 during randomized routing. At the end of the routing, when one of the nodes say p_2 decides to publish this entry into the DHT, invokes 3 *put* requests, namely, $put(H(madonna.jpg), [p_1, p_2, p_3])$, $put(H(warandpeace.pdf), [p_1, p_2, p_3])$, and $put(H(dangerous.mp3), [p_1, p_2, p_3])$. When a searcher performs search on any of these keys, he gets the same list of providers in the reply. He individually contacts each of the providers either sequentially or in parallel, where the authorization takes places before respective providers give access to the resource.

In the following section, we explain the process of resource anonymization in detail.

4.3 Resource Anonymization

When a resource r is selected for publishing, there exist two possibilities for anonymizing the resource: (i) anonymize the resource with a *random* set of keys (ii) anonymize the resource with a *fixed* set of keys. Both of these approaches are discussed in detail in the following section and referred as *anonymization by random selection* and *anonymization by fixed selection* respectively. We highlight the differences between both the approaches in the next section.

4.3.1 Anonymization by random selection (RS) approach

In this approach, the resource name ($r \in R$) is anonymized i.e., the resource name r is blended into $m-1$ number of other resource names randomly chosen from R. Then hash keys of all the m entries are published into the DHT. An adversary observing the resource keys of the (m, n)-entry may be able to derive the corresponding resource names of the $m-1$ keys (i.e., preimages) by employing a dictionary attack, which is considered as infeasible especially against some strong hash functions such as SHA-1. In spite of a successful dictionary attack, the adversary can not deduce which of these keys corresponds to a genuine resource.

When multiple providers try to publish the same resource r, since individual anonymization processes are independent, each may end up selecting non-overlapping random entries as part of anonymization. This can lead to resource privacy breach as explained later in Section 4.3.3. In contrast, the following anonymization technique forces all the providers to choose the same fixed set of entries for anonymizing r.

4.3.2 Anonymization by fixed selection (FS) approach

Instead of anonymizing the resource name by resource names picked randomly from a dictionary, the FS mechanism picks a fixed set of resource names for anonymization. It blends the resource key into $m-1$ number of keys[1] in such a way that the same keys are chosen every time the resource is published by different providers in the network. In order to achieve that the FS approach mandates that the same dictionary D must be used by all the providers in the system, in addition to selecting the keys for anonymization from the dictionary as specified in the following. To realize this, an *(m,n)-* entry with keys other than the ones present in the current dictionary will be dropped by peers during randomized forwarding. The FS- approach partitions the dictionary D into a set of equivalence classes, each of a fixed size m. Whenever a resource is picked up for publishing, it is anonymized with the members of the equivalence class the resource is part of. In the following, we discuss a mechanism to create such equivalence classes.

The dictionary D is arranged as a circularly ordered list of size 2^h with an h-bit index. The way the list is ordered is explained later. The dictionary supports the following operations related to retrieval of keys from the dictionary:

- $Lookup_D(k)$*:* returns the index $i \in [0, 2^h - 1]$ corresponding to the position of key k in the dictionary D.
- $getEntry_D(i)$*:* returns the key located at index i in the dictionary D.

Assume the dictionary D is represented by a circle as illustrated in Figure 4.3. The anonymization algorithm selects $\lceil \frac{m}{2} \rceil$ number of diameters on this circle. The keys located at indexes represented by the corresponding diameter end points, form the m number of keys required. The diameters are cleverly chosen in the following way: the *initiator diameter-* the diameter passing through the resource to be anonymized, is chosen first. Then $\lceil \frac{m}{2} \rceil - 1$ number of other diameters are chosen such a way they are uniformly spaced on the circle separated by a fixed angular distance $\theta = \frac{360^\circ}{\lceil \frac{m}{2} \rceil}$. As a result, at the end of anonymization process, there is no way to figure out the initiator diameter from which the other diameters are based on. Every diameter is equally likely to be an initiator. The algorithm is more formally explained here, which finally outputs the set S_m, a set of m keys:

Now we introduce the function $getIndex(i, \alpha)$ which outputs the index j at an angular distance of α from i on the above circle. If a h-bit identifier space resulting in a key space of 2^h size mapped onto the circle, thus resulting in an angular distance of $\frac{360^\circ}{2^h}$ between two successive points. Hence,

$$j = \left(i + \frac{\alpha}{\left(\frac{360^\circ}{2^h}\right)} \right) \mod 2^h$$

For example, in the Figure. 4.3, $j = i + 2^{h-1}$ as j is at an angular distance of 180° from i. If m is odd, one key from the resulting $m-1$ keys, is randomly picked and removed

[1]Note that hashes of these keys are published into the DHT.

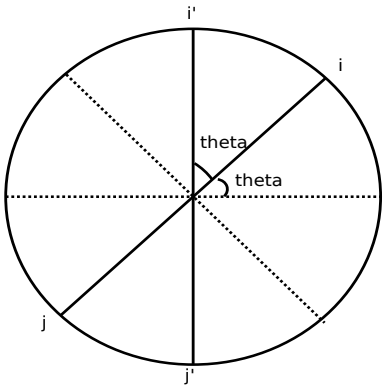


Figure 4.3: Circular approach for key anonymization

from the set[2].

4.3.2.1 Dictionary construction and distribution

In the following we discuss the issues related to dictionary construction and realizing FS approach for anonymization.

Algorithm 4.1: The circular approach

1: Let r be the resource to be anonymized and $i = Lookup_D(r)$
2: $S_m = \phi$
3: **while** $\mid S_m \mid < m$ **do**
4: $\quad j = getIndex(i, 180°)$
5: $\quad$ add $getEntry_D(i)$ and $getEntry_D(j)$ to S_m
6: $\quad i = getIndex(i, \theta)$ where $\theta = \frac{360°}{m}$
7: **end while**
8: **if** $\mid S_m \mid > m$ **then**
9: $\quad$ remove a member from S_m other than r randomly
10: **end if**

It is obvious from the FS description that, in order to have a fixed set of keys to be chosen by independent providers of the same resource, all the providers must be using the same dictionary. A static dictionary can be built into the PANACEA distributions installed on all the users in the system, if the corresponding resource sharing environment allows such a dictionary to be constructed a priori. In this case, all the potential resources to be published, should be known a priori, then the dictionary can cover all these resources.

However, in a typical resource sharing use case, predefining such a dictionary with all possible resource names is impractical. Hence, a dictionary construction mechanism is required which has to meet the following requirements:

1. *Supporting updates:* Dictionary must be expandable over time.

[2]The side effect of this on the resource privacy is not dealt in this thesis. An even value can be used for m in order to avoid this.

2. *Preserving diameter end points:* The keys falling in the indexes corresponding to diameters end points must not alter as the dictionary evolves over time.

In addition, we need a *dictionary distribution mechanism* in order to distribute the constructed dictionary to all the providers in the system.

In the following, we propose a simple dictionary distribution mechanism which assumes the presence of a single *dictionary manager*- a provider who performs the dictionary construction and distribution. This dictionary manager can be any well-known peer in the system or a web server whose credentials are built into PANACEA client distributions. It should be noted that the dictionary manager has no more knowledge than an unauthorized adversary, which will become explicit shortly, and hence, the PANACEA system achieves the same resource and provider privacies for the dictionary manager as that of an unauthorized user. The only intended requirement that the dictionary manager has to meet is, to implement the steps involved in the dictionary construction mechanism in a faithful way. Indeed, it is expected as the dictionary manager can not benefit w.r.t privacy by behaving otherwise.

The dictionary construction process involves the following steps:

1. Collecting *dictionary recommendations* from providers
2. *Committing* the recommendations to the dictionary
3. *Distributing* the updated dictionary to the providers

These steps are discussed elaborately here.

Dictionary recommendations: The providers in the system propose recommendations to the dictionary maintained by the dictionary manager. Each recommendation consists of a set of keys to be added to the current dictionary. We assume that this set of keys is chosen randomly from a provider's local dictionary or manually fed by the user through application interfaces. These keys need not be related to the resources the user wants to publish. Providers submit the recommendations at arbitrary points of time. However, recommendations can also be made when the resource the provider wants to publish, is not covered by the current dictionary.

The recommendation requests are submitted to the dictionary manager using anonymized routing like in Crowds [103]. This anonymous routing prevents the dictionary manager from attempting to breach provider privacy by finding the correlations between dictionary recommendations and later, the provider lists in the DHT, associated with the keys appearing in the recommendations.

As part of the anonymous routing, the intermediate peers can also contribute a set of keys to the recommendation request. Optionally, in order to prevent the intermediate peers learning from the key list present in the request, the keys can be encrypted with the public key of the dictionary manager.

In the evaluation, we assume the providers postpone the publication of a resource until the resource is covered by the current dictionary. However, providers need not

wait till their recommendations are committed and can publish a resource using RS-anonymization. We skip this step in our evaluation.

Committing recommendations: The dictionary manager accumulates all the recommendations and *commits* them to the dictionary by inserting the keys in the recommendations into the dictionary. This step may need to expand the current dictionary in size which must preserve the end points of diameters in the old dictionary as specified earlier. In the following, we propose one such committing mechanism.

The dictionary is first doubled in size by creating an equal number of free slots as the current dictionary size, with each free slot interleaved between two consecutive entries in the dictionary. Lets denote the new dictionary as D' and the original as D. It should be noted that entries in D' are identified with an index of size $h+1$ bits. Moreover, the altered positions of the keys in D in the new dictionary D' are given by

$$\forall k \in D, Lookup_{D'}(k) = 2 \cdot Lookup_D(k). \tag{4.1}$$

The dictionary manager, then places the keys from each recommendation randomly into the newly created free slots excluding the keys which are already present in the current dictionary. This random placement prevents a certain provider from controlling the diameter end points thus possibly compromising the resource privacy. For example, a malicious provider may target a particular resource and decide to compromise its resource privacy. Then he carefully crafts some bogus keys and submits a dictionary recommendation with all these keys. If all of these keys end up as diameters in the dictionary, other providers who publish the targeted resource inadvertently pick these bogus keys as part of the anonymization.

On the other hand, such a random placement may seem to aid the dictionary manager in compromising the resource privacy. He may correlate future *(m,n)-* publications with the keys in previous recommendations and make meaningful conclusions. However, keys used in an anonymization, (i.e., an *(m,n)-entry*) are from several different recommendations thanks to the random placement. Hence, the correlations can not differentiate one key from others.

Next, we have to verify that the diameter end points in D are kept intact in D'. Here, we prove that the angular distance between two keys in D remains the same in D' and thus does not make previous anonymizations done using D, obsolete. This also verifies the previous requirement on the diameter end points as angular distance between these two end points is 180.

Let us assume keys k_1 and k_2 are at an angular distance of α in D. Hence,

$$Lookup_D(k_2) = \left(Lookup_D(k_1) + \frac{\alpha}{\left(\frac{360^\circ}{2^h}\right)} \right) \bmod 2^h$$

Let this distance be α' in D'. Then,

$$Lookup_{D'}(k_2) = \left(Lookup_{D'}(k_1) + \frac{\alpha'}{\left(\frac{360^\circ}{2^{h+1}}\right)} \right) \bmod 2^{h+1}$$

From eq. (4.1), one can see that $\alpha' = \alpha$.

However, it requires a large number of recommendations to fill up the empty slots created as part of dictionary expansion. In practice, additional dictionaries (which are mutually disjoint) smaller in size can be used temporarily to commit the recommendations. These dictionaries must be merged into the main dictionary after sufficient number of keys are accumulated to fill the empty slots created in expansion of the main dictionary. Two equal sized dictionaries can be merged into a bigger dictionary twice in size much similar to the way a dictionary is expanded as explained above. Two dictionaries D_1 and D_2 of same size can be merged resulting in a dictionary D by interleaving each element of D_2 in between two consecutive elements of D_1, and as a result,

$$\forall k \in D_1, Lookup_D(k) = 2 \cdot Lookup_{D_1}(k), \text{ and}$$
$$\forall k \in D_2, Lookup_D(k) = 2 \cdot Lookup_{D_2}(k) + 1.$$

Upper bound on the number of dictionaries: As mentioned above, multiple dictionaries may exist depending on the frequency and number of input recommendations by the providers. At any instant, the maximum number of such dictionaries can be $h-1$.

Distribution: The dictionary can be initialized to null or to a large number of resource names from a number of different domains. Once a recommendation is received by the dictionary manager, it waits for a certain time period, expecting to accumulate further recommendations and commit all of them to the dictionary. All the recommendations received after this time window are processed similarly. The updated dictionary is lazily pulled by the publishers in the system, when needed. As long as the resource to be published by a provider is covered by the current version of the dictionary he stored locally, no dictionary fetch request is sent to the dictionary manager. Otherwise, the latest dictionary is retrieved from the dictionary manager.

We believe that the proposed simple solution does not result in scalability bottlenecks as dictionary updates will be infrequent after equilibrium is reached. The dictionary manager's unavailability for short durations can be tolerated. Finding more sophisticated solutions especially the ones employing multiple dictionary managers is a research challenge that needs to be investigated further.

4.3.3 Discussion

A resource r with multiple copies is expected to have a larger provider list than that of a resource published by only one provider. In the case of anonymization by the RS-approach, the provider list sizes of popular resources (resources with higher number of providers) tend to be noticeably larger than that of phantom keys as it is highly unlikely that the same phantom keys are chosen in independent random processes. An adversary can exploit this knowledge in order to infer the presence of genuine resources in the system. However, as long as a resource has a provider list size smaller or equal to the maximum possible provider list size for a phantom resource (denoted by n_f in the following sections), an adversary can not differentiate between a genuine and a phantom

resource solely based on the provider list sizes. Given this, any RS-mechanism must try to increase the value of n_f. We propose an extension to our basic approach that achieves this objective: a peer randomly selects a small partition of the set R denoted as R_L ($R_L \subset R$) and constantly employs R_L for the resource anonymization instead of R (referred to as *subset approach*).

The FS-approach aims to address the possible resource privacy breach for popular resources in the case of random key anonymization. It does so by making the providers choose for the same set of keys as part of anonymization of a certain resource. In other words, as mentioned earlier the circular approach partitions the resource key space into equivalence classes of size m. When a provider wants to publish a resource, all the members of the equivalence class the resource is part of, are selected for publishing as part of anonymization.

Hence, every key in the subset is as *popular* as any other key in the set, thus phantom keys will also have the provider list size as that of the most popular resource in the system. This is not the case in random resource anonymization, where the provider list sizes of phantom resources can not match some of the most popular resources and thus the popular resources can be easily inferred to exist in the system, thus breaching the resource privacy.

The FS-approach preserves the privacy of the resources at all levels of popularity. At each level, it blends a resource into a set of m number of resources which share the same level of popularity. In other words, the FS-approach maintains the popularity distribution of genuine and phantom resources in the system.

The FS-approach allows an adversary to construct all the (m, n)- entries where a resource appears by simply querying the DHT m number of times. However, this is not always possible because of the overlaps in values which prevent the adversary from constructing exact provider lists for individual *(m,n)*-entries. In the random approach, the adversary can not construct the *(m,n)*-entries since $m-1$ number of keys that could be associated with a key are chosen randomly. However, by looking at the provider lists of DHT entries, the adversary can attempt to construct such entries by picking the keys which have same provider set repeated. But the adversary has to query the system with a very large number of queries. Any case, having the knowledge of multiple *(m,n)*-entries does not breach the privacy as proved in the Theorem. 4.4.1.

The RS-approach allows a publisher to multiplex multiple resource publications into a single *(m,n)*-entry. As part of resource anonymization, he can select more than one genuine resource, thus publish multiple resources in a single entry, yet all resources having the same resource privacy. This is not possible in the FS-approach unless the provider owns and wishes to publish other resources corresponding to the entries in the fixed set.

In the rest of the chapter, we assume resource anonymization using FS-approach unless mentioned otherwise.

4.4 Privacy and Performance Analysis

In this section, we analytically study the privacy offered by *PANACEA* for adversaries with different capabilities for a particular resource r, using probabilistic and information theoretic approaches. We also estimate the expected communication overhead of our approach whenever it is appropriate. It must be noted that an adversary is unauthorized by default to access the resource r and thus its resource and provider privacies must be preserved.

First we analytically model the privacy properties of *PANACEA*. Then we highlight various possible adversary models that are relevant to the PANACEA system and quantify the privacy achieved by different adversaries.

4.4.1 Analytical modeling of privacy

We do the privacy analysis using two approaches: *probabilistic approach* studies the privacy properties on a probabilistic scale (introduced in Figure 2.2) and *information theoretic approach* studies the effectiveness of anonymization techniques in terms of entropy.

4.4.1.1 Probabilistic approach

We denote:

i) $P_{K,a}$ (resp. $P_{K,u}$) as the probability for an authorized (resp. unauthorized) user to deduce the existence of a certain genuine resource. We refer to this as *resource probability* in the rest of the chapter for ease of presentation.

ii) $P_{V,a}$ (resp. $P_{V,u}$) as the probability for an authorized (resp. unauthorized) user to deduce the provider of a certain resource. We refer to this as *provider probability* in the rest of the chapter.

iii) $P_{\_}$ as the probability for an authorized or unauthorized user to deduce the non-existence of a certain *non-existing* resource.

Definition An access-controlled system is said to provide *higher* privacy if it promises:

i) Lower probability for an unauthorized user to deduce a resource's presence and its provider ($P_{K,u}$, $P_{V,u}$)

ii) Lower probability for a user deducing a resource's non-existence ($P_{\_}$)

Under this definition of privacy, any *privacy-efficient* access control mechanism should aim to:

- Minimize $P_{K,u}$, $P_{V,u}$, $P_{\_}$, which should ideally be 0 as in unstructured P2P systems.

- Maximize search cost $C_{s,u}$ for unauthorized users and ideally close to that of the unstructured P2P systems.

However, the *search efficiency* of the privacy-enabling mechanism should remain high, i.e.,:

- $P_{K,a}$, $P_{V,a}$ should ideally be 1 (as in structured P2P systems), and
- The search communication cost $C_{s,a}$ should be kept low and ideally close to that of the structured P2P systems.

We express the privacy and search cost metrics of PANACEA in terms of the corresponding metrics of structured and unstructured P2P systems. To this end, we use superscripts U and S to denote metrics for unstructured and structured P2P systems respectively, and no superscript for PANACEA, e.g., $P^U_{K,u}$ refers to unstructured systems and the equivalent metric for PANACEA is $P_{K,u}$.

The analysis can be understood as a tool to help the system designer to analyze the posteriori probabilities achievable by unauthorized users possible for various values of system parameters in a minimality model of privacy breach, as explained below.

4.4.1.2 Information-theoretic approach

The privacy offered by systems employing anonymization can be quantified in terms of the effectiveness of the anonymization mechanisms, which is measured using an information theoretic approach. In [55, 113], such an approach was proposed to measure privacy employing *entropy* H as an anonymity metric, which is defined as:

$$H = -\sum_i p_i \log_2 p_i \,, \tag{4.2}$$

where p_i is the attacker's estimate of the probability that a participant i was responsible for some observed action. Entropy is maximized to $\log_2 |A|$ if equal probability is assigned to all members of the anonymity set A, and it is minimized to 0 when $|A| = 1$. According to [113], a system with entropy H has *effective anonymity set* of size 2^H.

As an adversary in PANACEA may have different information sets (i.e., each resulting from different observations), *conditional entropy* H_0 [55] is a more appropriate metric, which is given by:

$$H_0 = \sum_y Pr[Y = y]H(X|Y = y) = \sum_y E_y H(X|Y = y), \tag{4.3}$$

where X is a random variable of the private aspect to be preserved and Y models the different observations y. E_y denotes expectation with respect to observation y.

We study the privacy properties of the system by introducing two metrics for a resource r, namely *resource entropy* and *provider entropy*. They are represented as $H_{K,a}$ and $H_{V,a}$ respectevily, for the case of an authorized user.

Now we highlight various adversary models and quantify the privacy for each.

4.4.2 Adversary models

A curious adversary participates in the PANACEA publishing and searching process and tries to deduce about the presence of certain resources in the system and the corresponding providers. Such an adversary can do this either staying single or colluding with other adversaries. These are referred as a *single* adversary model and a *collusive group* adversaries model respectively.

In addition, an adversary can assume multiple roles in the system: as a *participant* in the randomized routing, a *host* node where an index entry is hosted in the DHT, and a *searcher*. Typically, an attacker can play some or all of these roles at the same time. In the case of the adversary as a searcher, we study the case of privacy breach possible for *non-interactive* and *interactive* query models, where in the former the privacy attack involves a *single* query from the user where as in the latter, *multiple* queries.

Unless specified otherwise, in all cases, we assume a *minimality attack* scenario [127]. The minimality attack model assumes an attacker being well versed with the technical details of the underlying system, namely for the case of PANACEA, the attacker is assumed to know the anonymization process, the values of m, n, and μ. For example, the attacker can be part of the system (by being a provider or a host or a participant in the routing) and thus is aware of the parameters configured by the system administrators.

We also discuss briefly, in Section 4.4.5, a special type of adversary in the form of an authorized user who has access to a particular resource with a particular provider of an *(m,n)*-entry. Similar to most of the privacy-preserving data sharing systems, PANACEA's provider privacy can be breached by a global adversary who can eavesdrop the communication channels and determine with certainty, who initiated publishing an *(m,n)*-entry. However, the resource privacy can not be breached even by such a powerful adversary.

We categorize all of the above adversary cases into two:

1. *Single Entry Case:* where, a *single (m,n)*-entry
2. *Multiple Entry Case:* where, a finite *set* of *(m,n)*-entries

involving the particular resource r under attack, are available to the adversaries. For clarity of presentation, they are referred as *SEC* and *MEC* in the rest of the draft.

In the context of the above adversary models, the *SEC* covers the single searcher non-interactive (i.e., single query) adversary model. The *MEC* involves all other possibilities where the adversary, either alone or colluding with others, can collect a set of *(m,n)*-entries, for example, a single adversary acting as a host or a single searcher with multiple searches on the index. An adversary, as a participant in randomized routing, can accumulate a finite number of (m, n)- entries. A host node of a key can also collect a set of (m, n)-entries. As mentioned earlier, it is possible for a searcher too to construct the set of (m, n)-entries that contributed to a particular entry in the DHT, but not always, as overlaps in the set of providers across entries prevent from doing so.

4.4.2.1 Notation

We use the following notation in our analysis. Let N be the number of peers in the system. As metioned already, multiple peers can own the same resource and publish into the system independently from others. For example, a popular movie file can be present with more than one provider in the system. We identify each presence of this resource as a *copy* and access rights on one copy are no way connected to that of another copy. A user may be authorized to a copy owned by a certain provider and unauthorized to a copy owned by a different provider. But when a user is authorized to copies of multiple providers, his resource request will be successful if any one of the authorized copies can be retrieved. Thus, the authorized copies are identical in satisfying a query for that resource by an authorized user. Let N_c be the expected number of copies of a genuine resource r and $N_a \leq N_c$ be the number of copies that a particular user is authorized to access. We call a user as *unauthorized* to r, when he is not authorized to access any of the N_c copies of r, i.e. $N_a = 0$.

4.4.3 Single Entry Case (*SEC*)

4.4.3.1 Probabilistic approach

Computation of $P_{V,a}$: First, we quantify provider privacy for an authorized user. There are three cases that can arise:

Case (i): If any of the N_a copies, where he is authorized to, was published to the DHT, he could deduce the provider of the resource with probability 1. The probability that at least one of N_a copies was published into the DHT is $1 - (1 - \mu)^{N_a}$.

Case (ii): On the other hand, consider the case that none of the N_a copies was published into the DHT (probability of which, is $(1 - \mu)^{N_a}$), but at least one of the remaining $N_c - N_a$ copies was published (probability of which is $(1 - (1 - \mu)^{N_c - N_a})$). In this case, the user first contacts all the providers associated with $H(r)$ and then floods the search request, where he deduces the provider with probability $P^U_{V,a}$. In case of unsuccessful flooding, the user tries to deduce the provider from the provider list present in the DHT. For time being, lets assume the case the user already established the existence of the resource. If l is the provider list size, it implies $\frac{l}{n}$ number of (m, n)-entries contributed to this resource's DHT entry. Hence, in this case, we claim that the provider is deduced with probability $\frac{l/n}{l} = \frac{1}{n}$ [3].

Now, lets assume the general case where the resource existence is not known to the user. In this case, in addition to finding a provider, the user has to decide on the genuinity of the resource. For one *(m,n)*-entry, this can be understood as picking one *(1,1)*-entry out of $m \cdot n$ number of *(1,1)*- entries. Hence, for a provider list of size l, we have to choose $\frac{l}{n}$ number of *(1,1)*-entries from a total of $l \cdot m$ number of entries, thus resulting in a provider privacy of $\frac{\frac{l}{n}}{l \cdot m} = \frac{1}{m \cdot n}$.

[3] after considering the probability of a valid provider being chosen randomly by another valid provider as part of publishing as negligible.

However, the provider list size need not grow in multiples of n for each insertion, because of "collisions" (i.e. provider id conflicts) in the provider lists across the multiple (m, n)-entries of the resource copies. We account for this effect with a *collision probability* f_v.

Case (iii): When a DHT entry is not found for the resource key (probability of which is $(1-\mu)^{N_c}$), the user attempts to deduce the provider by flooding ($P^U_{V,a}$). Hence,

$$\begin{aligned} P_{V,a} = &\left[1-(1-\mu)^{N_a}\right]\cdot 1 + (1-\mu)^{N_a}\cdot(1-(1-\mu)^{N_c-N_a})\cdot \\ &\cdot\left[P^U_{V,a}\cdot 1 + (1-P^U_{V,a})\cdot\frac{1}{m\cdot n(1-f_v)}\right] \\ &+ (1-\mu)^{N_c}\cdot P^U_{V,a} \end{aligned} \tag{4.4}$$

The above equation captures the fact that an authorized user, if fails to contact one of the N_a providers in the search process, can deduce about the possible provider with the same probability as an unauthorized user. Thus, the probability of deducing the provider for an authorized user is at least that of an unauthorized user.

Computation of $P_{K,a}$: We apply similar reasoning to formulate resource privacy $P_{K,a}$ for an authorized user with the following consideration for case (ii): In case of unsuccessful flooding, the user tries to deduce the resource existence from the DHT entry. Here, the user can assign only a probability of $\frac{1}{m}$ to the existence, as the queried key is mixed with $m-1$ other ones in the (m, n)-entry, in addition to what can also be deduced by flooding ($P^U_{K,a}$). Therefore,

$$\begin{aligned} P_{K,a} = &\left[1-(1-\mu)^{N_a}\right]\cdot 1 + (1-\mu)^{N_a}\cdot(1-(1-\mu)^{N_c-N_a})\cdot \\ &\left[P^U_{K,a}\cdot 1 + (1-P^U_{K,a})\cdot\frac{1}{m}\right] + (1-\mu)^{N_c}\cdot P^U_{K,a} \end{aligned} \tag{4.5}$$

Computation of $C_{s,a}$: Next, we quantify the search cost $C_{s,a}$ in terms of the number of nodes visited by the search query from an authorized user. First, a user searches in the DHT which incurs a cost of $C^S_{s,a}$. Thereafter, we account for two possible cases- none of N_c copies or some i copies of the resource are published into the DHT. The former case can happen with probability $(1-\mu)^{N_c}$ where the user employs flooding, incurring a cost of $C^U_{s,a}$. In the latter case, $i\cdot n$ number of providers are contacted. If none of them has an authorized copy (probability of which, is $(1-\frac{N_a}{N_c})^i$), the user employs flooding. Overall, $C_{s,a}$:

$$\begin{aligned} C_{s,a} = &C^S_{s,a} + (1-\mu)^{N_c}\cdot C^U_{s,a} + \sum_{i=1}^{N_c}\binom{N_c}{i}\mu^i(1-\mu)^{(N_c-i)}\cdot \\ &\left[i\cdot n\cdot(1-f_v) + \left(1-\frac{N_a}{N_c}\right)^i C^U_{s,a}\right] \end{aligned} \tag{4.6}$$

Equations for $P_{K,u}$, $P_{V,u}$, and $C_{s,u}$ can be derived from eq. (4.4) to eq. (4.6) by having $N_a = 0$ and replacing the terms $P^U_{V,a}$, $P^U_{K,a}$, $C^S_{s,a}$, $C^U_{s,a}$ by $P^U_{V,u}$, $P^U_{K,u}$, $C^S_{s,u}$, $C^U_{s,u}$ respectively.

Computation of P_{-} and $C_{s,-}$: Finally, we derive P_{-}, i.e. the probability to deduce the non-existence of a non-existing resource. Given an event space $\Omega = \{\text{DHT}, \neg\text{DHT}\}$ that a non-existent resource is found or not in the DHT respectively, P_{-} is given by:

$$\begin{aligned} P_{-} = Pr(- \mid \Omega) =& Pr(- \mid \neg\text{DHT}) \cdot Pr(\neg\text{DHT}) + \\ & Pr(- \mid \text{DHT}) \cdot Pr(\text{DHT}) \text{ , where} \\ Pr(- \mid \neg\text{DHT}) =& P^{U}_{K,u} = 0 \\ Pr(- \mid \text{DHT}) =& \frac{m-1}{m} \\ Pr(\text{DHT}) =& \left[1 - \left(1 - \frac{1}{\mid R \mid}\right)^{\mu N_r (m-1)}\right] \\ Pr(\neg\text{DHT}) =& 1 - Pr(\text{DHT}) \end{aligned} \tag{4.7}$$

N_r is the total number of resources in the system and R is the resource namespace. $Pr(- \mid \neg\text{DHT})$ expresses the probability that a resource is non-existent, given that it is not found in the DHT. This is similar to the probability of deducing the existence of an unauthorized resource for a user in unstructured P2P systems, because an existing resource is same as a non-existing resource for an unauthorized user. $Pr(- \mid \text{DHT})$ is the probability that the resource corresponding to the key does not exist. $Pr(\text{DHT})$ expresses the probability that a phantom resource from namespace R may have been inserted into the index, while $Pr(\neg\text{DHT})$ is the complement of $Pr(\text{DHT})$. Observe that P_{-} is minimal (~ 0) for reasonable values of the various parameters. Also, we estimate the expected query cost to deduce the non-existence. The user first searches for an index entry and then employs flooding, hence,

$$C_{s,-} = C^{S}_{s,-} + C^{U}_{s,-} \, . \tag{4.8}$$

4.4.3.2 Information-theoretic approach

Computation of $H_{V,a}$: First, we calculate the entropy of PANACEA for provider anonymity against an authorized searcher. Here, the random variable X models the publisher of the requested resource and the random variable Y models possible information sets observed by the searcher: (i) an authorized copy is found in the DHT, (ii) an unauthorized copy is found in the DHT and an authorized copy is found by flooding, (iii) an unauthorized copy is found in the DHT but no authorized copy is found by flooding, (iv) no copy is found in the DHT but an authorized copy was found by flooding, and (v) no copy is found in the DHT and no authorized copy was found by flooding. Note that, if unauthorized, no reply is returned to the searcher in flooding and thus the cases of finding an unauthorized copy in flooding do not exist. If the searcher has made the observations (i), (ii) or (iv), then the provider entropy is 0. In case of observation (iii), where an unauthorized copy of the resource is found in the DHT and no authorized copy was found by flooding, we calculate $H(X|Y = (iii))$ according to the following reasoning: $\mu(N_c - N_a)$ copies are expected to be published in the DHT resulting in a

provider list of size $l = \mu(N_c - N_a)n(1 - f_v)$. Therefore, the probability that a copy out of the $\mu(N_c - N_a)$ ones resides at one of these providers conditioned on the probability of resource's existence is $\frac{\mu(N_c-N_a)}{m \cdot l} = \frac{1}{mn(1-f_v)}$. Also, $(1-\mu)(N_c - N_a)$ copies are not published in the DHT and the probability that a copy resides at any other provider (i.e. apart from the l ones) is $\frac{(1-\mu)(N_c-N_a)}{N-l}$. However, since there exist $N_c - N_a$ copies in total in this case, the sizes of the aforementioned sets of l and $N - l$ number of providers are divided by $N_c - N_a$ to derive the effective anonymity set sizes. Finally, in the case of observation (v), where no copy is found in the DHT and no authorized copy was found by flooding, each peer in the set of N peers has a probability of $\frac{N_c}{N}$ for being a provider. Therefore, the provider entropy $H_{V,a}$, is given by:

$$\begin{aligned} H_{V,a} =& E_{\text{iii}} H(X|Y = \text{iii}) + E_{\text{v}} H(X|Y = \text{v}) \\ =& -(1-\mu)^{N_a}(1-(1-\mu)^{N_c-N_a})(1-P^U_{V,a})\cdot \\ & \Big[\frac{l}{N_c - N_a}\frac{1}{mn(1-f_v)} \\ & \log\left(\frac{1}{mn(1-f_v)}\right) + \\ & \frac{N-l}{N_c-N_a}\frac{(1-\mu)(N_c-N_a)}{N-l}\log\left(\frac{(1-\mu)(N_c-N_a)}{N-l}\right)\Big] \\ & -(1-\mu)^{N_c}(1-P^U_{V,a})\left(\frac{N}{N_c}\right)\left(\frac{N_c}{N}\right)\log\left(\frac{N_c}{N}\right) \end{aligned} \tag{4.9}$$

Computation of $H_{K,a}$: Next, we calculate the system entropy for resource anonymity. To this end, the random variable X models the existence of a resource, i.e., whether a resource name from the resource namespace R exists in the system or not. The random variable Y models the observations of the searcher for a requested resource as in the case of the provider entropy. The analysis follows a similar reasoning to the case of the provider entropy. Also, $l' = \mu(N_c - N_a)m$ is the expected number of keys in the DHT when $N_c - N_a$ copies may be published. Overall, the resource entropy $H_{K,a}$ is given by:

$$\begin{aligned} H_{K,a} =& E_{\text{iii}} H(X|Y = \text{iii}) + E_{\text{v}} H(X|Y = \text{v}) \\ =& -(1-\mu)^{N_a}(1-(1-\mu)^{N_c-N_a})(1-P^U_{K,a})\cdot \\ & \Big[\frac{l'}{N_c - N_a}\frac{1}{m}\log(\frac{1}{m}) + \frac{\mid R \mid -l'}{N_c - N_a}\cdot \\ & \frac{(1-\mu)(N_c-N_a)}{\mid R \mid -l'}\log\left(\frac{(1-\mu)(N_c-N_a)}{\mid R \mid -l'}\right)\Big] \\ & -(1-\mu)^{N_c}(1-P^U_{K,a})\log\left(\frac{N_c}{\mid R \mid}\right) \end{aligned} \tag{4.10}$$

The equations for $H_{V,u}$ and $H_{K,u}$ can be derived from eq. (4.9) and (4.10), respectively by replacing $N_a = 0$ and $P^U_{K,a}$, $P^U_{V,a}$ with $P^U_{K,u}$, $P^U_{V,u}$ respectively.

4.4.3.3 Resource anonymization with Random Selection

The PANACEA system's privacy properties in the case of resource anonymization with random selection (RS) approach are analysed in this section. As explained already in Section 4.3.3, the RS-approach has a unique case w.r.t resource privacy which is explained in the following. Consider the case of a very popular resource which many providers in the system own and are willing to share through the PANACEA system. In RS-approach, as the resource keys are selected randomly as part of anonymization, the same phantom key being chosen by many providers is less likely compared to the key corresponding to the popular resource. Hence, by observing the sizes of the provider lists with keys in the DHT, one can assume that the keys with longer lists are more probable to be genuine than not. However, we believe that its not trivial for an adversary to estimate the maximum provider list size a phantom key can take in the system. Yet, we assume the worst case where the adversary can estimate this value, which is modeled by a parameter n_f in the following analysis. In the rest of the section, we denote the case of RS-approach with superscript *RS*.

Computation of $P^{RS}_{K,a}$: This formulation is done in a similar way it is done for eq. (4.5) in addition to the following. If the size of provider list of the (m,n)-entry of the queried resource is larger than n_f, the user can deduce the existence of the resource in the system with probability 1. Otherwise, a probability $\frac{1}{m}$ can be assigned to the existence, as the queried key is mixed with $m-1$ other ones in the (m,n)-entry. We consider this observation for formulating the privacy properties.

Moreover, note hat collisions in the resource keys are also possible as providers choose key randomly. These collision are accounted for, with a probability f_k, which increase the expected provider list size of a key [4].

$$\begin{aligned} P^{RS}_{K,a} &= \left[1-(1-\mu)^{N_a}\right]\cdot 1 + (1-\mu)^{N_a}\cdot(1-(1-\mu)^{N_c-N_a})\cdot \\ &\quad \mathbf{I}(E(n)) + (1-\mu)^{N_c}\cdot P^{U}_{K,a}, \text{ where} \\ \mathbf{I}(l) &= \begin{cases} 1, & \text{for } l > n_f \\ P^{U}_{K,a}\cdot 1 + (1-P^{U}_{K,a})\cdot\frac{1}{m}, & \text{otherwise} \end{cases} \end{aligned} \tag{4.11}$$

$E(n) = \mu(N_c - N_a)n(1-f_v)(1+f_k)$ is the expected provider list size for the queried resource.

Computation of $P^{RS}_{V,a}$: The provider privacy for authorized user is computed as follows:

$$\begin{aligned} P^{RS}_{V,a} &= \left[1-(1-\mu)^{N_a}\right]\cdot 1 + (1-\mu)^{N_a}\cdot(1-(1-\mu)^{N_c-N_a})\cdot \\ &\quad \cdot\left[P^{U}_{V,a}\cdot 1 + (1-P^{U}_{V,a})\cdot \mathbf{I}''(\mathbf{E}(\mathbf{n}))\cdot\frac{1}{n(1-f_v)(1+f_k)}\right] \\ &\quad + (1-\mu)^{N_c}\cdot P^{U}_{V,a}, \text{ where} \\ \mathbf{I}''(l) &= \begin{cases} 1, \text{ for } l > n_f \\ \frac{1}{m}, \text{ otherwise} \end{cases} \end{aligned} \tag{4.12}$$

[4] In fact, a DHT entry can also be present for the key being a phantom one. However, we assume this probability as negligible, as $|R|$ is big compared to the number of genuine resources.

4.4.4 Multiple Entry Case (*MEC*)

The following theorem deals with the privacies offered by a set of *(m,n)*-entries and later we argue that, in general, this case exhibits privacy properties similar to that of the single *(m,n)*-entry case.

Theorem 4.4.1. For an adversary v_k, the offered resource and provider privacies in terms resource and provider probabilities, by an arbitrary set of (m, n)-entries $\mathtt{E}$ for a given resource r, respectively, are

- 1 and 1 if the set E contains $m \cdot n$ number of (m, n)-entries with complete overlap in their provider sets
- $\frac{1}{m}$ and $\frac{1}{m(n-o)}$ if E contains a very larger number of (m, n)-entries with o number of common providers
- $\frac{1}{m}$ and $\frac{1}{mn}$, otherwise.

Proof. We represent the set $\mathtt{E}$ as

$$\begin{aligned} \mathtt{E} &= \bigcup_j \{\mathtt{e}_j\}, \text{ where} \\ \mathtt{e}_j &= (K_j, V_j) \text{ is the } j^{\text{th}}\text{entry} \end{aligned} \tag{4.13}$$

The set $\mathtt{E}$ can be divided into two subsets $\mathtt{E}_1$ and $\mathtt{E}_2$ using the input resource r such that

$$\begin{aligned} \mathtt{E} &= \mathtt{E}_1 \cup \mathtt{E}_2, \text{ where} \\ \mathtt{E}_1 &= \bigcup_j \{\mathtt{e}_j\} : r \in K_j \\ \mathtt{E}_2 &= \bigcup_j \{\mathtt{e}_j\} : r \notin K_j \end{aligned}$$

For the user v_k, the set $\mathtt{E}_2$ can not reveal any information about resource r, hence, the rest of the discussion considers only the set $\mathtt{E}_1$.

As discussed earlier, the FS-approach anonymizes each resource with the same fixed set of other resources and hence, all $K_j \in E_1$ are identical. Given this, the possible cases that arise are:

1. *Case-I:* All of the sets V_j are identical.
2. *Case-II:* All of the sets V_j are mutually disjoint.
3. *Case-III:* A set of values repeat in multiple sets V_js.

Case-I: When the set V_js are the same, the adversary must be either a participant in the randomized routing or a host node in order to know about multiple identical *(m,n)*-entries. This case is particularly interesting when $\mid \mathtt{E}_1 \mid = m \cdot n$, which signifies that all the n providers listed in the entries own all the m resources listed. The resource

and provider privacies are completely breached in this case, thus resource and provider privacies are said to be 1.

Consider the case when $\mid \mathtt{E}_1 \mid \geq n \cdot (m-1)+1$. Out of all the possible odds, the best possible one for the adversary (i.e., the worst privacy offered by PANACEA) is that the set E_1 corresponds to $m-1$ resources being published by all n providers so far and the m^{th} resource which should be r, is about to start appearing. Hence, when $\mid \mathtt{E}_1 \mid$ is exactly $n \cdot (m-1)+1$, the adversary can be certain that r exists in the system and thus resulting in a resource privacy of 1. In this case, the provider privacy is $\frac{1}{n}$ and increases towards 1 in steps of $\frac{1}{n}$ for each additional entry in the set E_1.

Similar analysis can be done for the other cases where $\mid \mathtt{E}_1 \mid < n \cdot (m-1)+1$, but resource and provider privacy is not breached in this case.

Case-II: Since all V_js are mutually disjoint, there is no reason for the adversary to infer why one provider is more probable than the other, for being the valid provider in the list. Hence, no additional provider breach can occur because of availability of a set of *(m,n)*-entries ($\mathtt{E}_1$), and hence the provider privacy is said to be still $\frac{1}{mn}$.

We discuss the case of resource privacy here. When $\mid \mathtt{E}_1 \mid$ number of *(m,n)*- entries are observed by the adversary, he can take odds on different possibilities ranging from the case of $\mid \mathtt{E}_1 \mid$ number of providers publishing the same resource from the set of m resources $\{r_1, r_2, ..., r_m\}$ to the case of each resource being published in different possible multiplicities.

Consider the latter case, where resources are published in varied multiplicities by the providers. There are several possible events. Lets represent each event as a vector of size m, with i^{th} entry in the vector representing the number of occurrences of resource r_i in $\mid \mathtt{E}_1 \mid$ (denoted by e) publications. Hence, some of the possible events in this case are: $[0, 0, .., e]$, $[0, 0, .., 1, e-1]$, $[0, 0, .., 2, e-2]$, $[0, 0, .., 3, e-3]$, ..., $[e, 0, .., 0]$. This can be modeled as the arranging exactly e balls into m boxes where boxes are allowed to be empty. This can be done in $\binom{e+m-1}{e}$ number of ways, which is equivalent to the number of terms in the multinomial formula $(x_1 + x_2 + ... + x_m)^e$ where m and e are positive integers.

Out of $\binom{e+m-1}{e}$ events, there are $\binom{e+m-2}{e}$ events where a particular resource has 0 occurrences. In all the remaining events, the resource appears at least once, and hence one can attribute a resource privacy of $1 - \frac{\binom{e+m-2}{e}}{\binom{e+m-1}{e}}$.

In the former case where e number of publications of a single resource contributed to the observed list of (m, n)-entries. However, since the resource r is not differentiable with $m-1$ other resources, the resource privacy achieved would be $\frac{1}{m}$, irrespective of the size of the set E_1. Higher the size of this set and hence, higher the number of occurrences of same keys reveal only the popularity class of the resource as mentioned earlier, and not about their presence in the system.

But, while making recommendations to the dictionary, the providers blend a genuine resource into a set of resource names chosen randomly from a local dictionary, and hence, an anonymization set chosen in FS-approach, most likely has only one genuine resource.

The probability of having more than one genuine resource is negligibly small. On the other hand, because of the way the anonymization is done, one can not differentiate one resource over the other by just inspecting the contents of an *(m,n)*-entry. Hence, we argue that the system achieves a resource privacy of $\frac{1}{m}$ in this case too.

Case-III: Let's assume that a set $S_v \subset V$ (where $V = \bigcup_j V_j$ and $\mid S_v \mid < n$) appears in multiple, say l, number of V_js. Adversary as a searcher can not find out such overlaps. In this case, one can argue that the user v_k may try to attribute the resource r to the providers in S_v with a higher probability than to other providers. The provider privacy achieved from the above l number of *(m,n)*-entries is:

$$\frac{l}{m \cdot (l(n- \mid S_v \mid) + \mid S_v \mid)} = \frac{1}{m\,(n- \mid S_v \mid)}, \text{ for larger values of } l$$

For the maximum size of S_v in this case, which is $n-1$, the provider privacy as per the above formula will be $\frac{1}{m}$. The resource privacy is still being the same ($\frac{1}{m}$) as explained above. □

Discussion: Here, we discuss the case-I and III of above theorem in detail and argue that, in practice, probability of occurrence of such cases is negligible. Hence, in general, the PANACEA system at large, promises resource and provider privacies of $\frac{1}{m}$ and $\frac{1}{mn}$.

It must be noted that provider anonymization of PANACEA chooses a different set of providers for successive resource publications by an individual publisher. Thus, the size of the overlaps in providers across multiple *(m,n)*-entries is minimized. Hence, it is highly unlikely that a single participant in the randomized routing accumulates many *(m,n)*-entries with the same providers repeated for a given resource. Moreover, the way the *PStores* are populated with providers from passing *(m,n)*-entries can be attributed to the observed repetition of the same provider across *(m,n)*-entries.

Lets assume that there are N_c providers willing to publish a resource r. Assuming that all these providers have identical *PStores* of size N' (which is the worst possible case in this context), the probability that all of the N_c providers choose the same set of providers of size o for their *(m,n)*-entries is:

$$\frac{1}{\left[\binom{N'}{o}\binom{N'-o}{n-o}\right]^{N_c}}$$

Moreover, note that the probability of multiple *PStore*s having the same set of entries is negligibly small because of the way they are populated. Further, the privacy breach of case-I is limited to only the participants in the randomized routing and the host node for the resource key. For adversaries as searchers, the system behaves like the case-II.

4.4.4.1 Collusive group of adversaries

In *MEC*, there is a possibility that a group of peers in the system will collaborate among themselves in order to breach the privacy offered by the system. For example, when a

peer constructs an *(m,n)*-entry, if *n-1* number of providers listed collaborate with each other, finding the genuine provider becomes trivial for these colluders.

Users in a collusive group are assumed to share the following information with other group members: a provider appearing in an *(m,n)*-entry is due to whether his own publication event or a mere selection by some other provider as part of anonymization. In the latter case, the size of the anonymity set reduces by 1 for the colluders. Given this, the effective anonymity set size decreases by the number of members of a collusive group that appear in a single *(m,n)*-entry. It should be noted that if collusive groups are non-overlapping, there is no harm if members of two separate groups are included in a single *(m,n)*-entry, as there is no collusion among such groups. For example, if one member from each group is selected into the entry, the privacies offered will be still the same.

We assume the case of *static* collusive non-overlapping groups of maximum size c. We observe that typically collusive users are socially connected and share mutual trust which is limited to only the members inside the group. In general, such groups are limited in size as larger the group is, more difficult it would be to conceal the actions of the group members from the rest of the members in the system.

For $c = n - 1$, provider privacy can be breached successfully if and only if a provider chooses a particular set of $n - 1$ colluders for an *(m,n)*-entry. However, probability for this to occur is $\frac{1}{\binom{N'}{n-1}}$ which is ~ 0 for reasonable values of N' and n, where N' is the size of the *PStore*. Hence, as long as N' is significantly larger than c, a collusive group of adversaries can not breach provider privacy.

Note that this type of attack is concerned with only the provider privacy and resource privacy will not be breached by a collusive group. More over, the privacy breaches are limited to only members of such collusive groups. The system's privacy guarantees do not alter for other adversaries in the system.

4.4.5 More attack scenarios

In addition to the adversary models discussed in Section 4.4.2, PANACEA is subject to a special adversary who has authorization to some of the resources with some of the providers in an *(m,n)*-entry, thus has more knowledge of resources in consideration, than other adversaries. This knowledge may influence the privacies of other resources and providers appearing in a single *(m,n)*-entry. However, we argue that any possible privacy breach is negligible and does not hinder adaptability of PANACEA, for the following two reasons: First, a successful authorization with a provider for a resource appearing in the *(m,n)*-entry does not deny the possibility of existence of other resources in the system and the ownership of the current resource by other providers appearing in the entry. Secondly, considering the complete system at large, with all the resources and various attackers considered, such breaches can be tolerated given that they are limited to only these resources in consideration.

In addition, like against any other P2P system, the following illustrates some of the attacks possible on PANACEA which directly or indirectly compromise the privacy offered in the system. The solutions to these attacks in current literature can be applied to this case too and dealing with such attacks in detail is out of scope for the current work.

A *Sybil attacker* can gain multiple identities in the system and pretend to be multiple independent providers. Since all such identities essentially point to a single user, provider anonymization can be ineffective. However, wide variety of Sybil protection mechanisms available in the literature such as SybilGuard [129], can be applied in order to make make PANACEA Sybil-resistant.

Routing attacks are possible where malicious or rogue peers may not cooperate with the protocol of the system. In addition, PANACEA is subject to spurious publishing.

4.4.6 The state-of-the-art privacy preserving indexing mechanism

In order to compare the PANACEA system w.r.t the state-of-the-art solution in the literature for privacy preserving indexing of access controlled content, we consider the work introduced in [43] which is discussed in Section 2.4.3. This system is referred as *PPI* in the rest of the chapter. The PPI approach deals with provider privacy only. It partitions the set of providers into several groups and resources published by a single provider are mapped to his/her entire groups so that the provider can not be distinguished from others. The static groups in PPI are analogous to the dynamic anonymity provider set used in PANACEA.

Here, we model the privacy and search performance of the PPI system using the terminology introduced in Section 4.4.1.

4.4.6.1 Probabilistic approach

The PPI approach does not protect the resource privacy and an authorized user can always find the corresponding resource. Given c- the size of the privacy group and σ- the selectivity of the resource ($= \frac{N_c}{N}$),

$$\begin{aligned} P_{K,a}^{PPI} =& P_{K,u}^{PPI} = 1 \\ P_{V,a}^{PPI} = 1 \text{ and } & P_{V,u}^{PPI} = \frac{\sigma N}{N\left(1-(1-\sigma)^c\right)} \\ C_{s,a}^{PPI} =& C_{s,u}^{PPI} = N\left(1-(1-\sigma)^c\right) \end{aligned} \tag{4.14}$$

4.4.6.2 Information theoretic approach

$$\begin{aligned} H_{K,a}^{PPI} &= H_{K,u}^{PPI} = 0 \\ H_{V,a}^{PPI} &= 0 \\ H_{V,u}^{PPI} &= -\frac{N\left(1-(1-\sigma)^c\right)\frac{\sigma N}{N(1-(1-\sigma)^c)}\log_2\frac{\sigma N}{N(1-(1-\sigma)^c)}}{\sigma N} \end{aligned} \tag{4.15}$$

4.5 Evaluation

In this section, we evaluate the PANACEA system employing the FS-anonymization mentioned in Section 4.3.2. We present the results observed from simulation experiments and from a prototype deployment on PlanetLab described in detail separately in the following sections. The privacy and search metrics are studied as the system parameter μ is varied. In both the cases, we compute the parameters $P_{V,a}$ and $P_{V,u}$ as follows:

- $P_{V,a} = 1$ if a user is able to contact a publisher where he is authorized to access the requested resource.
- If the resource key is found in the DHT, then $P_{V,a} = P_{V,u} = \frac{1}{mn}$.
- Otherwise, if the resource key is not found in the DHT and no authorized copy is located by flooding, then $P_{V,a} = P_{V,u} = 0$.

The PANACEA's parameters are chosen as follows: key list size $m = 4$, value list size $n = 4$ and forwarding probability $\lambda = 0.6$.

4.5.1 Simulation based evaluation

The main objective of the evaluation is to prove the correctness of the analytical modeling of the PANACEA system's privacy and search efficiency presented in Section. 4.4.3 and study the performance of the system in a more general setting. In addition, we quantitatively position the PANACEA system w.r.t the state-of-the-art solution in the literature for privacy preserving indexing of access controlled content introduced in [43] discussed in Section 4.4.6 and Section 2.4.3. This system is referred as *PPI* in the rest of the chapter. We considered the group size in the PPI case same as the value of n in PANACEA. A qualitative comparison is presented in Section 2.4.3 and Section 4.6.

We implemented the PANACEA and PPI simulators in Java. The PPI simulator implements the system presented in [43]. We assume $N = 10000$ peers that use the system both to publish and search for resources. The *PStore*s on the peers are initialized with 25 random entries. The providers are organized in a Kademlia-like structured topology, but they are also connected over an unstructured overlay power-law network with average degree 7.5 and maximum degree 150. We conducted two types of simulation experiments, which differ in their resource distributions and the type of the generated queries. Each resource considered, is randomly assigned a publisher peer and a list of user peers who are authorized to access the resource. Any other peer is said to be an unauthorized user for this resource and publisher pair. Also, $ttl = 4$ was employed for limited-hop flooding in the unstructured overlay.

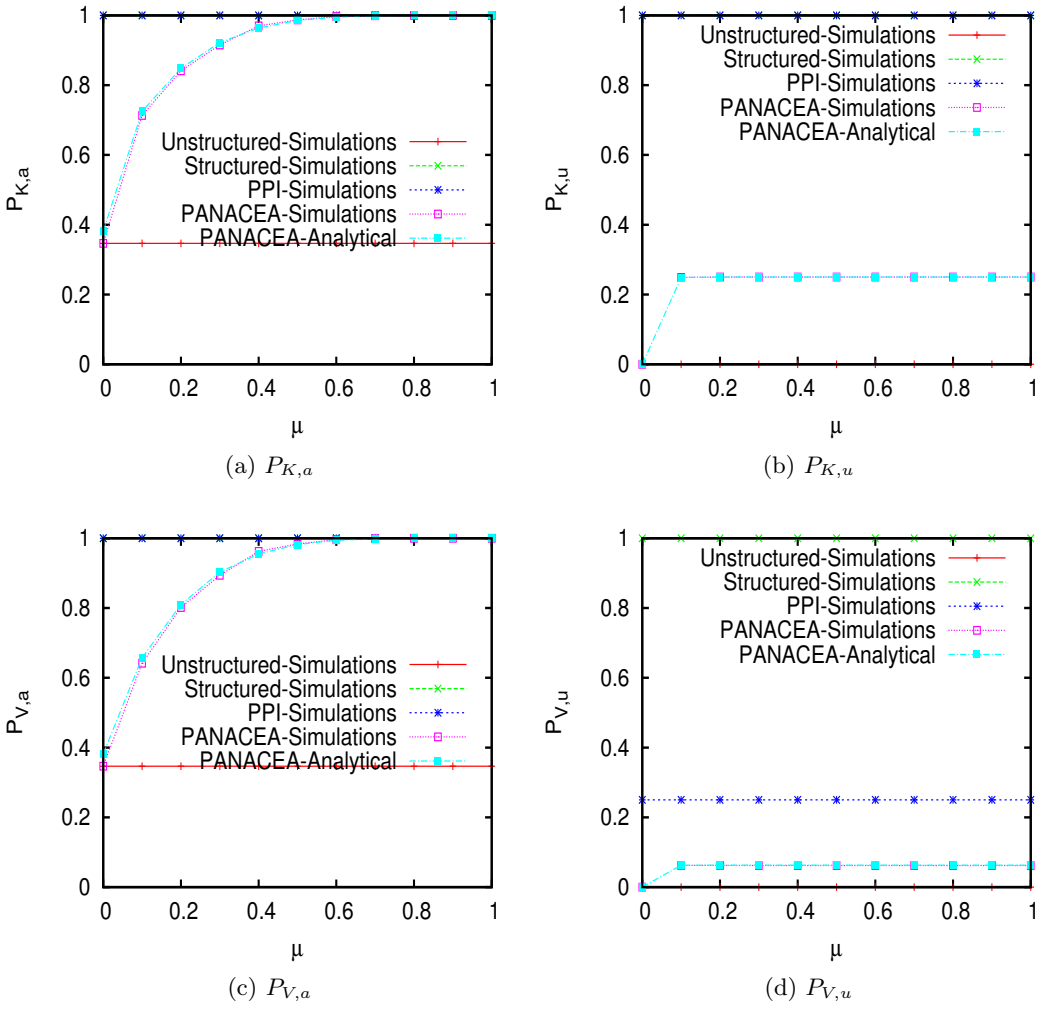
We run a total of 3 different experiments on the simulator and measure the resource, provider privacies including the search cost in all experiments described in the following.

4.5.1.1 Privacy and search cost

Experiment-1: Initially, we aim to verify the correctness of eqs. (4.4)- (4.6) using the simulation results with a rather static setting regarding resource popularity. Specifically, we assume 100 resources with $N_c = 50$ copies for each (thus $5K$ resources in total). 100 peers are randomly selected, each of which is inserted into the authorization list of random $N_a = 5$ copies. Each resource is then given to randomly chosen peers that publish them using the PANACEA publishing mechanism. The collision probability for provider lists is experimentally found to be $f_v = 0.002$. We also experimentally found that by searching in the unstructured overlay $P^U_{V,a} = 0.38$, $P_{V,u} = P_{\_} = 0$, a total of 912 distinct nodes are visited and 1687 messages are sent per query on the average. In order to measure $P_{K,a}$ and $C_{s,a}$, we generate authorized searches from the above 100 authorized peers for all of the 100 resources, thus $10K$ search queries in total. Also, in order to measure $P_{K,u}$ and $C_{s,u}$, we randomly select 100 unauthorized users that query the system for the same 100 resources. These experiments have been run 10 times each and the mean values are plotted in Figures 4.4, 4.5[5]. As depicted in these figures, the analytical equations model the privacy properties of the simulated PANACEA system very accurately, thus verifying the analysis presented in Section 4.4.3. As the probability of publishing μ increases, PANACEA approaches the search efficiency of a structured system (see Figures 4.4a, 4.4c). Note that for only $N_a = 5$ authorized copies in the system, a small value of $\mu = 0.6$ makes the search efficiency of PANACEA close to that of structured systems. The PPI approach as mentioned already, does not protect resource privacy and thus $P_{K,a} = P_{K,u} = 1$ as shown in Figures 4.4a, 4.4b. For authorized users, PANACEA shows performance similar to PPI after $\mu = 0.6$ but significantly improves resource privacy for unauthorized users (maximum $P_{K,u}$ is only 0.25, as show in Figure 4.4b, compared to 1 achieved by PPI). Thanks to the provider's existence conditioned on the resource existence, PANACEA enhances provider privacy over PPI as shown in Figure 4.4d where PPI offers a flat privacy of 0.25 and and PANACEA closer to that of unstructured systems. Therefore, PANACEA design meets its privacy objectives introduced in Section 4.4. For $\mu = 0$, $P_{V,u} = 0$. From $\mu = 0.1$ onwards, a provider list is found in the DHT for the queried resource, resulting in a constant privacy value of $\frac{1}{mn(1-f_v)}$.

In Figure 4.5a, the effect of μ on the search communication cost for authorized users is depicted. As μ increases, the probability to find a provider where the user is authorized also increases. After $\mu = 0.6$, there is no more search cost improvement, because the size of the provider list of the queried resource slightly increases. As shown later, in general, the search cost significantly decreases as μ increases. However, as observed from Figure 4.5b, the search cost for unauthorized users significantly increases (over the cost of limited-hop flooding) with μ, which is a highly desirable property of PANACEA. PPI shows a flat search cost equal to the selectivity of the resource times the group size.

[5]The privacy properties of PPI are analytically modeled in Section 4.4.6 and are skipped from the plots for brevity. The analytical model plots overlapped with that of PPI simulations.



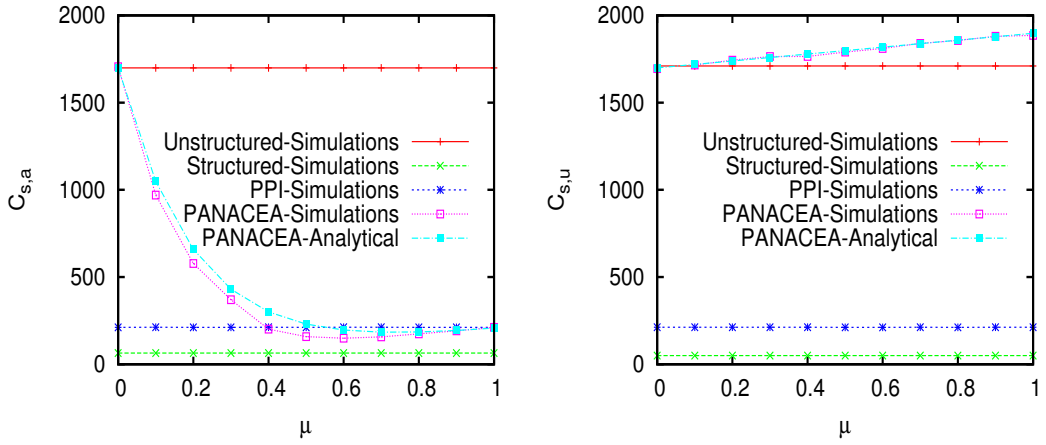
(a) $P_{K,a}$ (b) $P_{K,u}$

(c) $P_{V,a}$ (d) $P_{V,u}$

Figure 4.4: Privacy performance (Experiment-1)

Experiment-2: Next, we evaluate the privacy and the search performance of PANACEA for authorized and unauthorized users in a more general setting, where $10K$ resources whose popularity (N_c) follows Zipf distribution are published in the system. The maximum number of resource copies is 150 and their mean is 10 thus resulting in a total of $100K$ resources. A random number of 5 to 50 peers are chosen to be authorized to each resource. Each resource of the $100K$ ones is randomly assigned to a peer that initiates PANACEA publishing. This case represents the plots labeled $PANACEA\_random$ emphasizing the uniform random distribution of the parameter N_a. In Experiment-3, the value of N_a is a fixed fraction of N_c unlike the case of $PANACEA\_random$ which chooses N_a randomly. We study the system for two different cases where N_a is 10% and 20% of N_c (represented by $PANACEA\_0.10$ and $PANACEA\_0.20$ respectively in the plots).

In both cases, we randomly generated $20K$ number of authorized and unauthorized search queries separately. Again, the experiments are repeated 10 times and mean values



(a) $C_{s,a}$ (b) $C_{s,u}$

Figure 4.5: Search performance (Experiment-1)

of the results are plotted in Figures 4.6, 4.7.

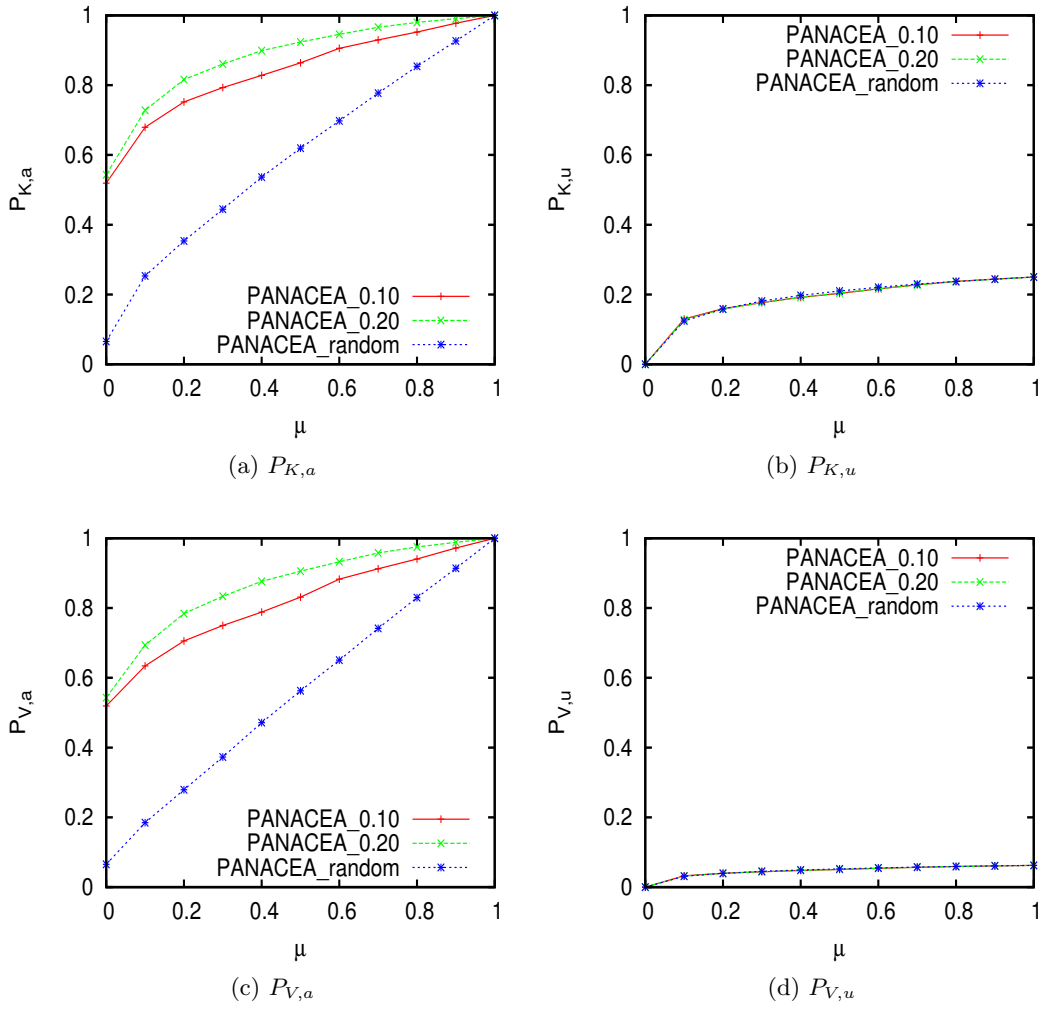
For $PANACEA\_random$, as depicted in Figures 4.6a, 4.6c, the search efficiency increases with μ for an authorized user reaching maximum at $\mu = 1$. For $PANACEA\_0.10$ and $PANACEA\_0.20$, the $P_{K,a}$ and $P_{V,a}$ show performance similar to that of Experiment-1 (Figures 4.4a, 4.4c). As $\frac{N_a}{N_c}$ increases, PANACEA's performance moves closer to that of the ideal case of $P_{K,a} = P_{V,a} = 1$. Note that the performance shown is average of $20K$ queries chosen randomly not considering their N_a and N_c values unlike in the case of Experiment-1. However, all the cases ensure a minimal resource and provider privacy for unauthorized users as shown in Figures 4.6b, 4.6d. Also, Figure 4.7a depicts that the search cost for authorized users decreases with μ, as opposed to that of unauthorized users as shown in Figure 4.7b.

4.5.1.2 System entropy

We demonstrate the entropy analysis discussed in Section 4.4.3.2 in Figure 4.8 with parameters $N_a = 5, N_c = 10$. For $\mu = 0$, the entropy $H_{K,u} \approx 18$ which is roughly equivalent to an anonymity set of size $\approx 200K (= \frac{|R|}{N_c})$. As μ increases, the entropy decreases due to the resource presence in the DHT. For an authorized user, the entropy gradually reduces with μ and becomes zero after $\mu = 0.6$. The provider entropy is demonstrated in Figure 4.8b. $H_{V,u} \approx 10$ $(=N_c/N)$ for $\mu = 0$. After $\mu = 0.6$, the provider entropy becomes zero for an authorized user. At $\mu = 1$, for an unauthorized user, the anonymity set size would be $n = 4$, which is verified by the plot as the entropy is 2 as shown in Figure 4.8b.

4.5.1.3 Anonymization by the RS-approach

In addition to the FS-approach based resource anonymization, we implemented the RS-based approach discussed in Section 4.3.1 in our simulator.



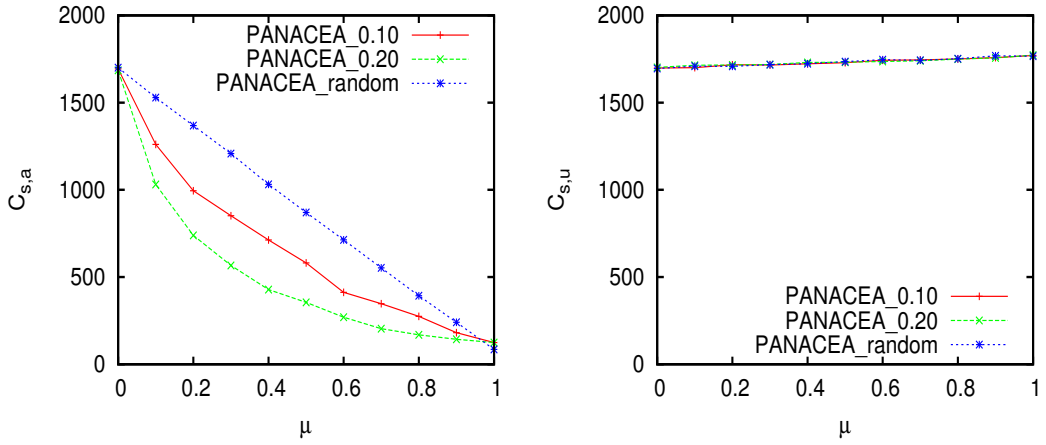
(a) $P_{K,a}$ (b) $P_{K,u}$

(c) $P_{V,a}$ (d) $P_{V,u}$

Figure 4.6: Privacy performance (Experiment-2&3)

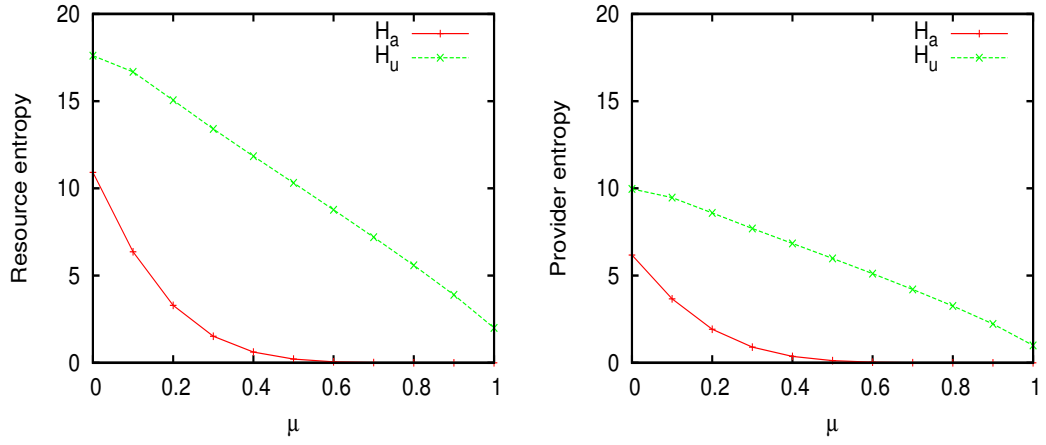
As mentioned in Section 4.3.3, the resource privacy may be breached by observing larger sizes of the provider lists. In this section, we will explore this experimentally and illustrate in terms of provider list sizes of genuine and phantom resources. However, since very popular resources are highly probable to be present in any data sharing system, we believe that the inability of the RS-approach to protect the resource privacy of the same does not hinder its adaptability.

We observe that as long as an existing resource has a provider list size less or equal to that of a phantom resource, an adversary can not differentiate between them. We assume again $100K$ resources Zipf-distributed with a mean of 10 and a maximum of 150 copies (as in above Experiment-2). For the *subset approach* (Section. 4.3.3) with $|R_L| = 25$, we observed that phantom keys have provider lists longer than those of the 87.6 percentile of the existing resources in the DHT (for $\mu = 1$). For this percentile of resources, the resource privacy for unauthorized users is $P_{K,u} = \frac{1}{m \cdot n}$ and for the authorized users is $P_{K,a} = 1$ for $\mu = 1$. For more popular resources, the adversary can exploit the provider



(a) $C_{s,a}$ (b) $C_{s,u}$

Figure 4.7: Search performance (Experiment-2&3)



(a) H_K (b) H_V

Figure 4.8: Entropy performance

list sizes to conclude their existence with some provider in the system.

Finally, by numerically evaluating eqs. (4.7) and (4.8) with $|R| = 2M$, $N_r = 100K$, $\mu = 1$, we observed the PANACEA system meets its design objectives in this case as well, as $P_- \sim P_-^U$ and $C_{s,-} \sim C_{s,-}^U$.

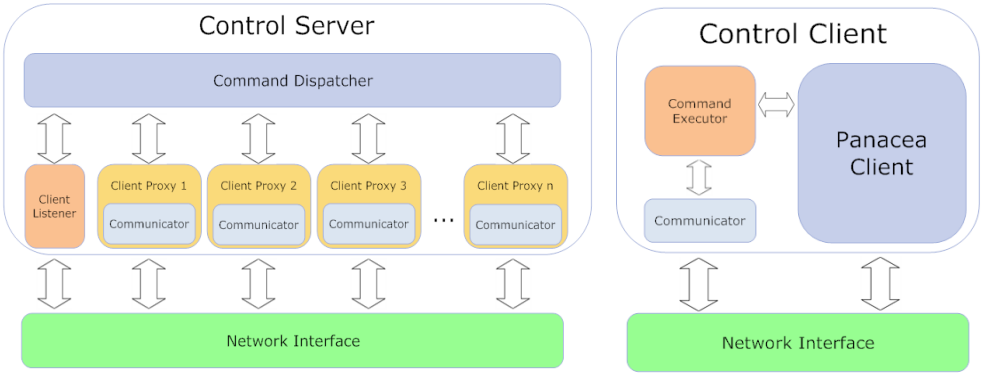
4.5.1.4 Discussion

Out of different types of experiments shown above, the Experiment-3 makes a more realistic choice as a resource becomes more popular (thus N_c increases), one can expect a reasonable proportionate increase in the number of providers a user is authorized with (N_a). The results shown for this case are promising and prove that PANACEA is a viable solution to meet both conflicting goals of privacy preservation and search optimization.

However, as seen in the results, the performance for the authorized user can be improved further (and hence $P_{K,a} = P_{V,a} = 1$, by choosing a value for μ independently for different resources in the system, which is in accordance with its popularity (N_c). For example, the search efficiency can be increased by choosing a high value for μ for unpopular resources. We envision that the publisher can approximately anticipate the popularity value of a resource based on either type of resource or past search experiences and choose appropriate value for μ.

In addition, in systems where privacy of non-existence of resources is not mandatory, the value of μ can be set to 1 without breach of any privacy.

4.5.2 PANACEA prototype deployed over PlanetLab



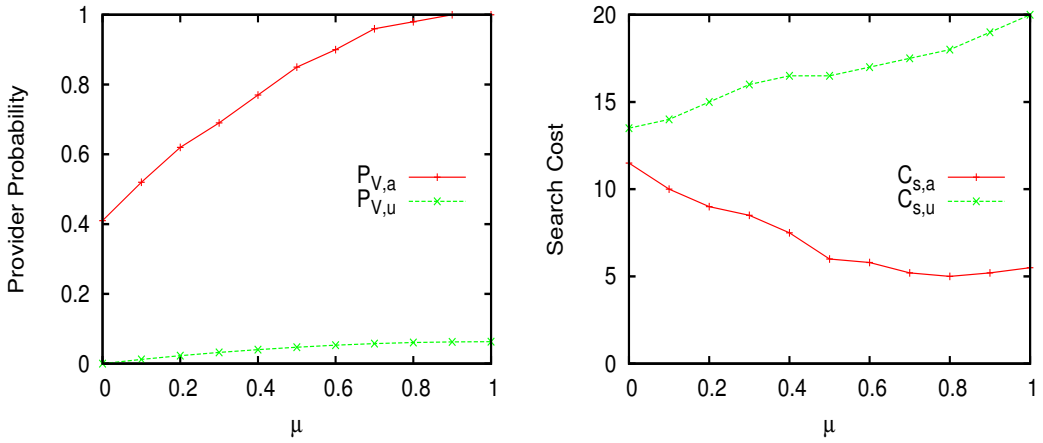
(a) Control Server (b) Client

Figure 4.9: PANACEA prototype architecture

We prototyped the PANACEA system with the RS-approach for resource anonymization, as a proof of concept. This prototype is built by embedding PANACEA into the open-source JKad client [20] which is a Java implementation for the popular Kademlia P2P network [89]. Other popular clients for the Kademlia network include eMule [21]. Thus, the resulting PANACEA clients are fully compatible with native JKad clients. We deployed the prototype on PlanetLab [22] which is a world-wide distributed infrastructure and currently consists of 1141 nodes at 545 sites spread across the world. For the sake of experimental study, we built a *Central Controller* which invokes various events on remote PANACEA clients deployed on PlanetLab nodes. The architecture of the prototype is shown in Figure 4.9.

We performed basic evaluation of PANACEA on this prototype deployment. The selected test scenario involves 50 PANACEA clients deployed over an equal number of PlanetLab nodes. We chose values for different parameters as follows: $N_c = 2$, $N_a = 2$, $m = 4$, and $n = 4$. Flooding is done with a degree of 2 and a ttl of 3. Figure 4.10a plots the provider privacy quantified in terms of provider probability both for authorized and unauthorized users and the search cost performance is shown in Figure 4.10b. These

initial experiments suggest that PANACEA system deployed, matches the performance behavior of PANACEA simulations presented in Section 4.5.1. We skipped other plots for brevity.



(a) Provider Probability (b) Search Cost

Figure 4.10: Evaluation on PlanetLab

4.6 Related Work

In Freenet [48], resource identifiers are generated in several cryptographic ways and inserted into the system based on these identifiers. It achieves access control, the resource, and provider privacies using cryptographic techniques, which however, involve complicated cryptographic key distribution and management overhead. Furthermore, resource discovery is not guaranteed and involves significant search communication overhead compared to structured systems. In addition, the searchers have to be associated with the providers *a priori*, in order to be informed about the cryptographic keys used to encrypt the content accessible to them. Instead, in our approach, search efficiency is high and new searchers can be dynamically authorized by providers to access the resources.

To enable access control in P2P systems, PHera [49] proposes a fine-grained access control framework based on super-peer-based P2P overlays where the access-control policies of sub-peers are enforced by the super-peers. Super-peers index the data of sub-peers and they could preserve data privacy by not replying to the queries from unauthorized peers. However, this approach assumes that all super-peers are unanimously trusted by their sub-peers to enforce their data privacy and access control policies, which is difficult to choose [42]. In PANACEA, peers can share their resources through an index hosted on untrusted nodes, and yet, can exercise access control.

Regarding the privacy of access-controlled content, a privacy-preserving approach for centralized indexing of such data is proposed in [42], which was discussed in detail in Section 2.4.3. A group of data providers arranged into a circle, iteratively circulate

a bloom filter representing the content hosted on the providers, bits of which are set probabilistically by the proposed algorithm. At the end of this iterative process, the index, represented by the bloom filter, emerges, which preserves data privacy regarding its location (i.e., provider privacy). However, as opposed to PANACEA, [42] does not address resource privacy. Furthermore, new resources can be easily inserted into the index of PANACEA, while index reconstruction is required in [42]. Even the provider privacy is breached if a succeeding and preceding provider collude, which can be overcome by mechanisms involving encryption mechanisms.

Privacy preserving data publishing is addressed widely in the literature especially in the database community [62], which include number of anonymization techniques that remove/blur the sensitive data from database records before releasing for publishing. The authors in [40] propose a privacy preserving framework that makes information released unusable to illegitimate users. Privacy preserving complete document indexing is addressed in [130] which builds a distributed indexing infrastructure which allows faster search of the documents without leaking information about other documents in the system.

The Crowds [103] and the Tor [35] systems offer anonymity for users activity on the web using randomized forwarding used by PANACEA. There exist a large number of works in the area of anonymous P2P systems that achieve publisher (source) or reader (searcher) anonymity or both [48, 117]. Additionally, the anonymity of a node hosting an index entry (resource) is also considered [56]. Community maintained dictionaries are used in [70] to enable privacy preserving data sharing in the context of online social networks. These dictionaries contain one-to-one mappings between values of certain attributes in an OSN profile. An attribute in the profile (e.g., name) is replaced with a mapped value (resp., a friend's name) so that an adversary sees only this replaced value. However, a legitimate user in the community can find out the actual value of the attribute by accessing the community dictionary.

4.7 Conclusion

In this chapter, we have proposed PANACEA, an indexing mechanism that enables multiple independent providers to publish their sensitive content whose access is controlled by appropriate user-defined access control policies. The index built, promises tunable resource and provider privacy for the content, yet with a very high search efficiency for the authorized users. We have analytically derived the privacy and search efficiency properties of the system employing probabilistic and information-theoretic approaches. Our analysis was verified by simulation experiments. We analytically and experimentally showed that PANACEA meets its design objectives. We employed the PANACEA mechanism in a Kademlia client and studied the privacy achieved in a real testbed. PANACEA matches the privacy efficiency of the best privacy-protecting solutions and the search efficiency of scalable overlay designs offering very high lookup efficiency.

Part IV

Privacy-aware Online Social Networking

Chapter 5

Is Decentralized Social Networking Feasible?

Just as the vast Internet opened up the world to everyone, it also opened up everyone to the world. The price paid is the privacy.

Gary Kovacs, Mozilla Corporation.

5.1 Introduction

The unprecedented success of Online Social Network (OSN) applications, such as Facebook, Twitter, etc., has resulted in a vast amount of personal information being available online. This information, on one hand, is of a great business value to the service provider, e.g., for personalizing ads, but on the other hand, makes the users vulnerable to privacy breaches and malicious exploitation, e.g., burglars locating vacant houses. Involuntary personal information leakage happens to a great extent in today's online social networks, thus breaching users' privacy [80] and causing a number of undesirable effects [79]. Several proposals exist in the literature that aim to increase the user privacy on OSNs without altering the existing social network infrastructures, by obfuscating [70] or encrypting [85] the profile content. Alternatively, semi or fully-decentralized OSN infrastructures such as, Peerson [45] and Diaspora [19] are also pursued. However, for such decentralized solutions to become viable alternatives for today's massive OSN infrastructures, several feasibility checks need to be performed regarding the performance of such systems. Performance metrics that uniquely characterize such OSN workloads need to be identified. To the best of our knowledge, no prior work has done a thorough empirical study of the various system properties of decentralized OSNs and the parameters that influence them. In this chapter, we experimentally explore these trade-offs and analyze the challenges towards the realization of decentralized OSN applications using

data traces from two real social networks Facebook and Twitter. We first define key efficiency metrics of such systems, namely availability, availability-on-demand, update propagation delay and replication degree. The first metric expresses the profile availability where as the second metric captures the profile availability upon request by friends in the network, the third metric expresses the data freshness, while the fourth metric quantifies the degree of replication done to achieve certain availability. To be explained later, an important parameter that affects all these metrics, is the *online time* of the user- the time period(s) within the day when the user is online in the network through his decentralized social network application client.

Since privacy is a serious concern in decentralized OSNs, in this work, we explore the case where profile replicas are placed only on trusted friend nodes in the social network, as opposed to a general peer-to-peer system, which replicates on any arbitrary nodes. Furthermore, decentralized OSNs that are built only on Friend-to-Friend (F2F) networks do not necessitate any complicated encryption mechanisms for data management. Employing different replica selection schemes and different realistic models to approximate users online times from Facebook and Twitter datasets, we experimentally establish that i) in order to achieve acceptable availability of profiles, a certain replication degree has to be met, ii) there is a trade-off between data availability, the data freshness, and degree of replication, iii) the number of replicas and their placement choice significantly affect the OSN's performance efficiency.

The rest of the chapter is organized as follows: Section 5.2 introduces various efficiency metrics for decentralized OSNs. In Section 5.3, we deal with the replica placement strategies. Experimental methodology is discussed in Section 5.4 followed by the results in Section 5.4.4. Literature work related to the current problem is presented in Sections 5.5. Section 5.6 concludes the chapter.

5.2 The Context

A well-designed decentralized OSN application should promise user experience and functionality similar to that of the existing centralized OSNs. A typical OSN allows its users to post messages or content onto their profiles (like the "wall" in Facebook) or on other people's walls, send personal messages, chat with online friends, discover new friends, and retrieve a feed of updates on friends profiles etc. In addition, the user should receive updates of the activities on his profile by his friends while he is offline. To this end, profile replication should be employed to keep the profiles available even when the owner users are offline in the system. As we explain later, the *online time* of users is an important parameter of the system that significantly affects profile availability.

5.2.1 Online Time Connectivity

Let OT_u denote the online time period of a user u. This is a continuous/discrete time period, with a predefined granularity (e.g., minutes, hours), during which the user is

active on the network and contributes bandwidth, storage, etc. through his OSN client. This parameter can be either a user input to the client or approximated by the client from the user's online history (for example, as shown in the later part of the chapter).

Let NG_u be the set of his friends (i.e., neighbors) in the social graph. Assume that the profile of a user u is replicated at some friends $R_u \subseteq NG_u$. In this study, we assume all friends of a user to be trusted for hosting the user's profile replica. This allows us to explore the best case performance of F2F based decentralized online social networks. The issues of breach of trust by friend nodes and node compromise by malicious attackers are skipped in this chapter and are briefly discussed in the next chapter.

The profile of a user u is *accessible* by an arbitrary user v if only if $\exists\ j \in R_u$ such that $OT_v \cap OT_j \neq \emptyset$. i.e., the user v and replica j must be *connected in time*. Hence, the replicas in R_u, can be either connected or disconnected in time. In the former case (referred to as *ConRep*), each replica of the user u's profile should overlap in time with at least one other replica, i.e., $\forall\ i \in R_u$, $\exists\ j \in R_u$ such that $OT_i \cap OT_j \neq \emptyset$. In the latter case (referred to as *UnconRep*), there exists at least one replica that is not connected in time to any of the remaining replicas. In such case, the replicas have to communicate among themselves using a third-party storage or a content delivery network (CDN). It may involve additional overhead due to encryption of the profile content in order to keep it secure from the malicious access. A decentralized OSN inherently privacy- conscious, should adopt the *ConRep* approach for the replica selection as the social profile content is exposed to only trusted friends.

5.2.2 Technical Requirements

For the decentralized OSN platforms to become viable alternatives to centralized siblings, a number of technical requirements need to be realized, which are discussed below:

5.2.2.1 Storage requirements

The profile of a user should be highly available regardless of the user's own connectivity to the system, which can be achieved by profile replication. In order for all the friends of a user to eventually access the user's activity in the OSN, all the updates should be communicated across all the replicas with certain guarantee on data consistency. We believe that a requirement of *eventual consistency* would be adequate for decentralized OSNs. The issue of consistency is explained in detail in next chapter when a storage architecture for a decentralized social network is introduced. In addition, the replica selection should ensure *fairness* among the replicas by balancing the storage and communication overhead involved in hosting a replica uniformly. Another requirement concerning the *data freshness* requires that any updates on a user's profile should be accessible by all his friends as soon as possible, with an upper bound on the delays incurred in reaching consistency, especially when the replicas are not online always.

5.2.2.2 Privacy requirements

Typically in a privacy-aware OSN, a user's profile is configured to be accessible only by the 1-hop friends in the network. Hence, the replication mechanism should be optimized to increase the availability of the profile to the 1-hop friends. Since delegation of the profile access control to other nodes (even trusted nodes) is a cause of concern due to a potential privacy breach to the profile, the degree of replication should be minimized. Storing the user profiles in encrypted form on untrusted nodes may be needed to improve availability, but it involves complicated key management and distribution, especially to enforce access control on the profile content.

5.2.3 Efficiency Metrics

In the following, we define several performance metrics for measuring the efficiency of decentralized OSNs.

5.2.3.1 *Availability*

The fraction of time in a day, a user's profile is accessible through the replicas which serve the profile during their online times. All the user nodes where a profile is replicated, contribute to the profile availability during their online times. Note that maximum achievable availability for a certain user is limited by the union of the online times of his friends in an F2F model. The availability of a user (say u)'s profile is the fraction of sum of his replicas online times (i.e., $\bigcup_{v \in R_u} OT_v$) over length of a day (i.e., 24 hours). For example, a user's profile is said to be 50% available if all his replicas together are online for 12 hours a day.

5.2.3.2 *Availability-on-Demand*

This metric quantifies the accessibility of the profile for only the friends of a user. Since in a privacy-friendly OSN, typically a user's profile is mostly configured to be accessible to only 1-hop friends in the network, we model *availability-on-demand* as the availability of the user profile whenever it is actually accessed by his 1-hop friends. This metric expresses the *effective* profile availability based on the activities of friends. We introduce two variations of the metric:

Availability-on-Demand-Time: Fraction of the union of the online times of the friends of the user, the profile is available through the replicas. It should be noted that these friends are expected to access the profile during their online time, by definition. Formally, this metric for a user u can be quantified as $\frac{\bigcup_{v \in R_u} OT_v}{\bigcup_{v' \in N_G(u)} OT_{v'}}$.

Availability-on-Demand-Activity: Fraction of the times there was an activity on a user's profile in the past and the profile was available. Note that a user's profile may be accessed by his friends at arbitrary points of time during their online times. This metric captures availability of the user's profile whenever it was actually accessed by his friends as elaborated later in Section 5.4.

5.2.3.3 *Update Propagation Delay*

The delay between the occurrence of an update event at a certain replica of a user and its arrival on another replica is the update propagation delay between these two replicas. This delay depends on the length of the online time overlap among the replica points as shown in the Figure 5.1. In the case of connected replicas (*ConRep*), a weighted replica time-connectivity graph is computed with the replicas as the nodes and edges between two replicas if they are connected in time. The weight of each edge set to the update propagation delay between the two end nodes. Updates among replicas are propagated via the multi-hop shortest path on this graph through other replicas.

We explain the calculation of this delay in the example of Figure 5.1. We assume three replicas of a certain user's (say user v) profile residing at nodes v_1, v_2, and v_3 with different continuous online times represented with begin (t_s) and end (t_e) times as $OT_{v_1} = [t_s^{(v_1)}, t_e^{(v_1)}]$, $OT_{v_2} = [t_s^{(v_2)}, t_e^{(v_2)}]$, $OT_{v_3} = [t_s^{(v_3)}, t_e^{(v_3)}]$, for which the replica time connectivity graph is also shown in the figure. Let an update event happen at replica v_1 at time t. Then, this update would be communicated to v_2 at time t', which would take $24 - d_1$ hours, where d_1 is the number of overlapping hours between v_1 and v_2. Furthermore, since at time t' node v_3 is not online, in order for the update to reach the replica v_3, it would take an additional $24 - d_2$ hours, where d_2 is the gap between t' and $t_s^{(v_2)}$ in hours. Thus, in total the update propagation delay between v_1 and v_3 would take $48 - d_1 - d_2$ hours, which is the worst possible case for communicating a profile replica update at node v_1 to node v_3.

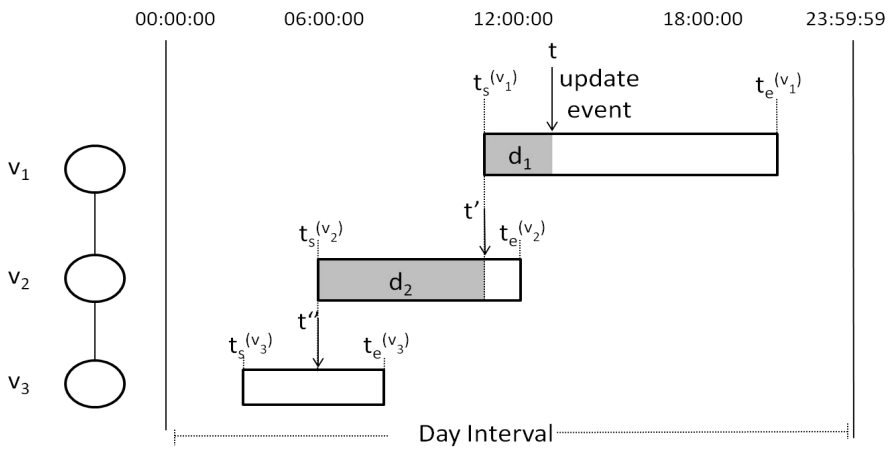


Figure 5.1: Illustration of the replica time-connectivity graph and the propagation of update from replica v_1 to v_3.

The *Update Propagation Delay* for a user is the maximum of propagation delays between all pairs of the replicas. It is the weight of the longest of the shortest paths among all pairs of replicas in the above weighted replica time connectivity graph. Hence, in the above example, the update propagation delay for the user v is $48 - d_1 - d_2$ hours. This metric captures the maximum/worst case update propagation delay for transferring

updates among replicas of a given user profile. This metric directly impacts the data freshness.

The Update Propagation Delay has two aspects to be considered: one, the end-to-end delay as explained above and second, the actual delay as *observed* by a user (friend) who can experience the delay only in relation to his online time window i.e., the time when the friend is offline should be excluded from the above update propagation time. To this end, we refer the former delay as the *actual* and the latter as the *observed* propagation delay. In the above example, the observed delay for the node v_3 is $t'' - t_s^{(v_3)}$ whereas the actual delay is $48 - d_1 - d_2$ hours.

5.2.3.4 *Replication Degree*

It is the number of replicas hosting a user's profile. Higher the replication, better the availability a user's profile has. On the other hand, this metric expresses the storage and communication overhead involved in replicating the user's profile. Moreover, it can be seen as the degree of potential privacy breach of the profile, which can occur with or without the replica host node being aware of the breach. Higher the replication degree, more is the level of potential exposure of personal information to others. An extremely privacy-conscious user wants to ideally have a replication degree of 0 for his profile.

5.3 Replica Selection Policies

In F2F storage model, the maximum replication degree can be equal to the number of friends a user has. However, it is desirable that the replication degree should be low because of overhead associated with replication. In order to choose a set of replica points for a user's profile from all of the user's social network friends, we employ various criteria as described in detail in the following:

5.3.1 Maximizing the availability (*MaxAv*)

In this approach, we choose as replica locations the user friends, which maximize the availability of the user profile. Since each user's online time is known a priori, the maximum availability achievable for a user u in a F2F model is bounded by $\mid \cup_{f \in NG_u} OT_f \mid$. Hence, the replica selection algorithm should choose the minimum number of replicas-/friends that jointly achieve this availability. We model this problem as the conventional *set cover problem* with the set to be covered (the *universe*) chosen as $\cup_{f \in NG_u} OT_f$. The online hours of the friends (OT_f) represent the family of the subsets of the universe in the set cover problem. Since finding an optimal solution for the set cover problem is NP-hard, we solve the problem in a greedy way that chooses replicas incrementally until no improvement is observed in the achieved availability. The algorithm, at each step, chooses the friend who is online for the highest number of remaining uncovered hours in the universe.

In the *ConRep* case, at each step of the greedy heuristic, while choosing the next replica/friend, only the friends which are connected in time to any of the already chosen replicas, must be considered. Out of all such overlapped friends, the one whose online time has the least overlap with the current covered set, is chosen as the replica.

5.3.2 Maximizing the availability on demand (*MaxAvD*)

Here, the friends, the union of whose online times maximizes the availability-on-demand (Section 5.2.3.2) are chosen as the replicas. In *maximizing the availability-on-demand w.r.t time (MaxAvDt)* approach, replicas are chosen with an objective of maximizing the availability-on-demand w.r.t time (described in Section 5.2.3.2). In *maximizing the availability-on-demand w.r.t activity (MaxAvDa)*, availability-on-demand w.r.t activity is optimized. In both the cases, the problem is modeled a as set cover problem as done in the case of *MaxAv* with the universe sets defined as follows: for *MaxAvDt* case, the union of online times of the user's friends ($\cup_{f \in NG_u} OT_f$) forms the universe whereas for the *MaxAvDa* case, it is the set of times during which some activity was observed on the user's profile.

5.3.3 Most active friends as replicas (*MostActive*)

This approach prioritizes the friends who perform the most activity on a user's profile, for placing the replicas. The intuition is to improve the availability of the profile to the friends who need/access it the most. As a side-effect, the availability-on-demand will be maximized. The top-k most active friends where the activity is measured as the number of times interaction happened between the user and his friend are chosen as replicas.

5.3.4 Random friends (*Random*)

In this approach, friends of the user are randomly chosen to place the profile replicas.

In the *ConRep* case, the replicas should be connected in time for all of the above replication strategies. This set is computed iteratively, with the set initialized to the user himself. A new replica is added to this set if and only if it overlaps in time with any of the replicas in the set. The iterative algorithm halts after all possible replicas are considered.

5.4 Experimental Methodology

In this section, we describe the methodology we used for the experimental analysis of the performance trade-offs of decentralized OSNs w.r.t different replica placement policies described in the Section 5.3, based on real data traces from Facebook and Twitter.

5.4.1 Dataset description

For our study, we needed social networks datasets which include i) the social graph, ii) the user activities happened among the users and iii) the timestamp of each activity, which helps to approximate online times as explained below. Most of the datasets in the literature lack at least one of these requirements. We employed two datasets that meet our needs: a Facebook [124] and a Twitter dataset [64], which are explained in detail in the following. The user degree distribution of both the datasets is presented in Figure 5.2, which is the number of friends (resp. followers) in the social network Facebook (resp. Twitter).

The activity considered was the wall posts (for Facebook dataset) and the user's tweets (for Twitter dataset). We believe that considering even richer set of activities like passive profile viewing, personal communication or chats will not alter the experimental methodology and the mechanisms used in the algorithms. In addition, more types of activities chosen will enhance the performance of the algorithms w.r.t the metrics. For example, in *Sporadic* model explained below, an extra activity would increase the user's online time and thus availability of his profile.

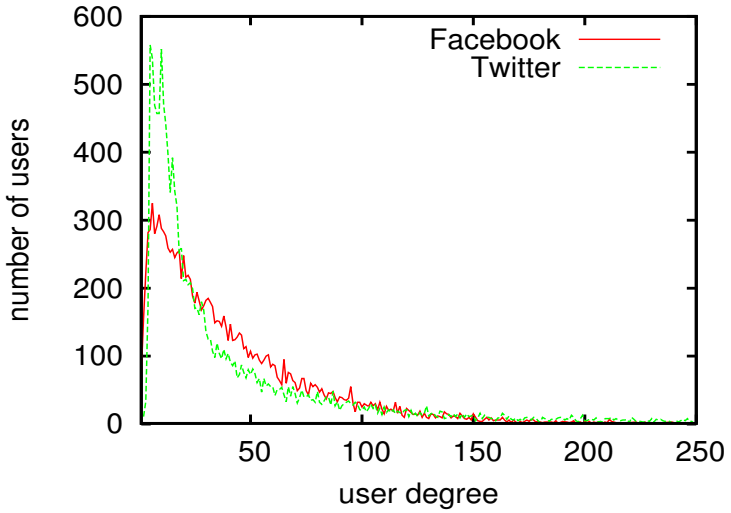


Figure 5.2: User degree distribution of the Facebook and Twitter datasets.

5.4.1.1 Facebook

The Facebook dataset employed is the NewOrleans Network dataset [124], which has a total of 63,731 users creating a total of 876,994 wall-posts. A wall-post in the dataset has a receiver, a creator, and a timestamp.

In a decentralized Facebook, a user's profile is accessed (by his friends) from any of the profile replicas which are online at that instant.

5.4.1.2 Twitter

We employed a simplified version of the Twitter dataset of [64], which originally included 158,324 tweets made by a total of 23,162 users in Twitter between 10-Sep-2009 and 24-Sep-2009. From this dataset, we excluded all the users whose followers are not present in the dataset. A tweet in the dataset has a receiver, a creator, and a timestamp, similar to a wall-post described before.

In a decentralized Twitter, we chose to replicate a user's profile on his followers. This is a natural choice as the majority of the information flow in Twitter is from the user to his followers. When a user is offline, his replicas are used by his followers to access his tweets and by his followees (users followed by him) to communicate their tweets to him.

We filtered out users with very little activity (less than 10 wall-posts or tweets) from the above datasets. We ended up with a total number of 13884 users for Facebook, with the average degree 41 (i.e., number of friends) and an average number of 50 activities per user. For Twitter, the filtered dataset contains 14,933 users with average degree of 76 (i.e., number of followers).

5.4.2 Simulation

We built a Java-based simulator that processes the Facebook and Twitter datasets and computes the profile replication points for the users in either dataset according to the replica placement algorithms of Section 5.3. Then, all the user activities in the datasets are replayed in the system and the efficiency is measured in terms of the metrics specified in Section 5.2.3, as the replication degree is varied. The user online times which are not directly available in the datasets, are approximated by applying different models as explained in the next subsection.

In each case, the replication degree is varied from 0 (i.e., only the user stores his profile) to the maximum limit: the number of friends/followers of the user. In both the *ConRep* and *UnconRep* cases, for a user, some of his friends may have online times which do not overlap with any of the replicas. It should be noted that the number of such disconnected friends is indirectly reflected in the *availability-on-demand-time*. The *Maximizing Availability on Demand (MaxAvD)* approach- (refer Section. 5.3.2) based replica placement is not presented in the results as in the F2F model, the *MaxAv* and *MaxAvD* exhibit similar behavior. In case of *MostActive*, if there are no sufficient number of friends with non-zero activity, random friends are chosen as replicas.

In the case of most active friends as replicas (*MostActive*), a friend who created most of a user's received activity (in the activity dataset) is considered as the most active friend.

Once the user online times are computed, only some part of the user activity in the datasets falls within this online time (we term it as *expected* activity) while the remaining falls outside (termed as *unexpected* activity). The metric availability-on-demand-activity (shown in the plots) captures availability of profiles for both the activities together.

Availability of user profile for unexpected activity will have positive effect on the users of the system as they perceive the system to be available even when it is not expected, as per the definition.

5.4.3 User online time models

As mentioned earlier, users online time is an important metric that affects the performance. However, approximating the same from the known datasets, is a challenge in itself and we model the online times based on user activities in three different ways:

5.4.3.1 Sporadic

This model assumes that user is online in the OSN several times a day sporadically, and each appearance can be seen as a *session*. We consider sessions of fixed length with each user activity performed at a random point in the corresponding session duration. We make use of the user online behavior studies in [44, 111] to model the session lengths. As found for Orkut in [44], most active users stay online for more than an hour in a session, while 22% of the sessions last less than 20 minutes. The study in [111] found an average session length of 40 minutes for the case of Facebook, with some sessions even lasting for several days. To this end, we employed a fixed session length of 20 minutes, as a conservative choice for both the Facebook and Twitter studies. Unless specified otherwise, *Sporadic* refers to a 20 minute session length. In addition, we explore in detail, the effect of the session length on the performance metrics for different session lengths for the case of Facebook dataset.

5.4.3.2 Continuous-Fixed Length

In this model, all the users in the network are assumed to be online, each day of the week, during a continuous time window of a fixed length (we chose 2, 4, 6 and 8 hours as the duration lengths). The actual time-of-day for each user is centered around the majority of their activity times as per the datasets. The intuition behind this model is that users stay online for continuous time periods in which they perform activities arbitrarily, as observed for Skype [69]. This model is denoted as $FixedLength$ in the results (Section 5.4.4).

5.4.3.3 Continuous-Random Length

This is same as the above model, except that each user randomly chooses his own length of the online time window from the range $[2, 8]$ hours. This is denoted as $RandomLength$ in the results.

5.4.4 Limitations

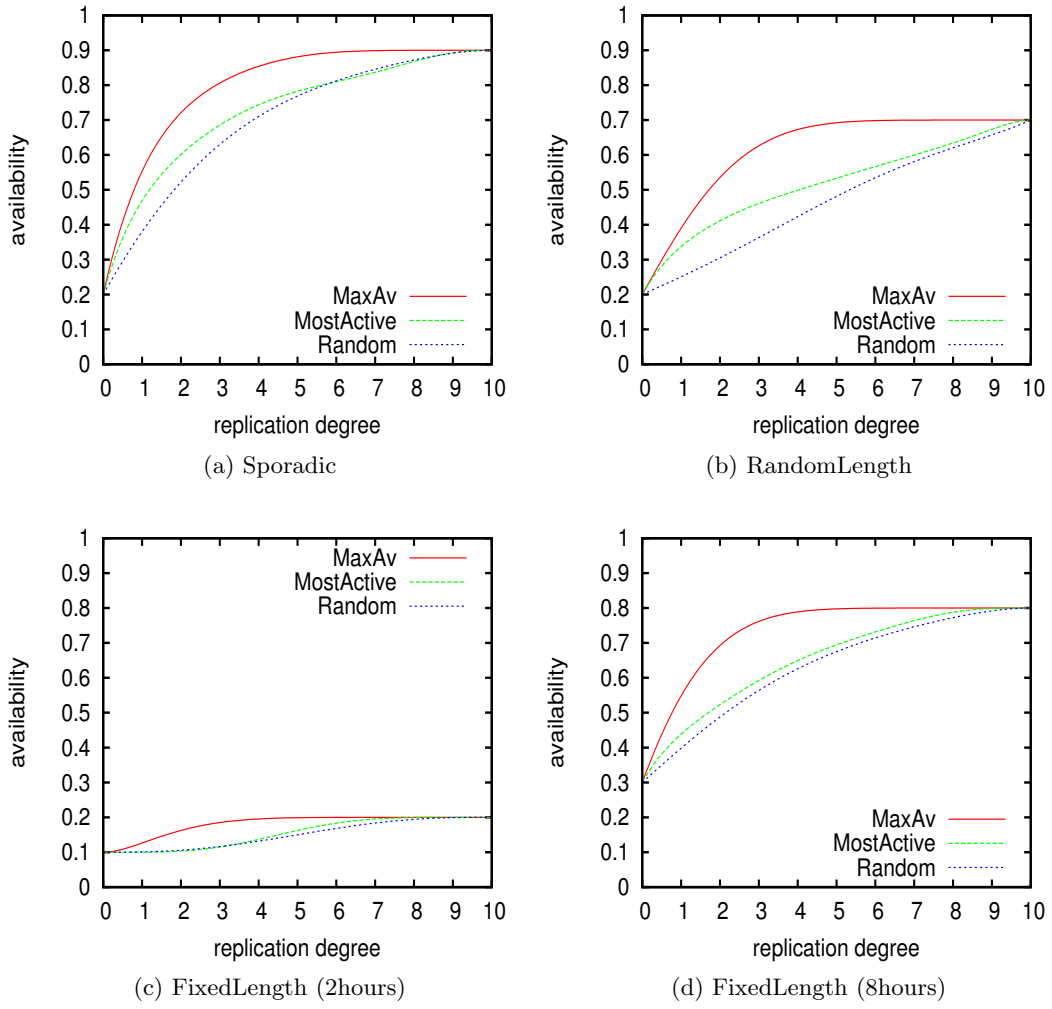
As with any empirical studies, our results and conclusions are, invariably limited by the limitations and inconsistencies of the datasets we choose. First, we consider only one form of activities among users in the social network: wall-posts (Facebook) and tweets (Twitter). Second, as already mentioned, online times of the users are not included in the datasets, and are approximated by different models, as explained in 5.4.3. Nevertheless, we believe that, since the considered activities constitute the majority of the overall activities in Facebook and Twitter [126], our datasets can be considered representative for obtaining results of general applicability. In this section, we illustrate the results of our empirical study for the Facebook and Twitter datasets, in terms of the efficiency metrics as the replication degree varied for all online time models of Section 5.4.3. Unless specified otherwise, we present, the averaged results for the users with a particular degree and we chose degree 10, as both the datasets have the most number of users (Facebook:$\sim$ 300 and Twitter: $\sim$ 550) with this degree. Hence, replication degree is varied from 0 to 10. For the *FixedLength*, only the 2hour and 8hour online duration cases are presented, for brevity. Experiments involving randomness, i.e. *Random* placement or *RandomLength* model, are repeated 5 times and averages are presented. Availability is computed as the fraction of number of distinct online hours (resp. minutes for *Sporadic*) of replicas over a day i.e., 24 hours (resp. 1440 minutes), while the availability-on-demand-time is the fraction of number of distinct online hours of replicas over that of all his friends together.

5.4.5 Facebook

5.4.5.1 Availability vs. Replication Degree

Availability increases with replication degree as is illustrated in Figure 5.3 and Figure 5.4 for the cases of connected and unconnected replicas respectively. As expected, the *MaxAv* replication scheme outperforms others, while achievable availability stabilizes after replication degree 6, 5, 4 for the online time models *Sporadic*, *FixedLength* and *RandomLength* respectively. *MostActive* replication is better than the naïve *Random* placement and achieves the availability of *MaxAv*, but with a higher number of replicas being used. Also, observe that achievable availability for *FixedLength* for 2 hours case is very low.

Note that the actual number of replicas chosen may be much lower than the maximum allowed replication degree in *ConRep* case, as enough connected replicas can not be always found. However, for *UnconRep* case, the achievable availability is higher as expected, since the replica locations can be selected regardless of their online time connectivity. This can be seen in Figure 5.4a and 5.4b for the *FixedLength* case. The plots for other online time models are skipped for brevity reason.



(a) Sporadic

(b) RandomLength

(c) FixedLength (2hours)

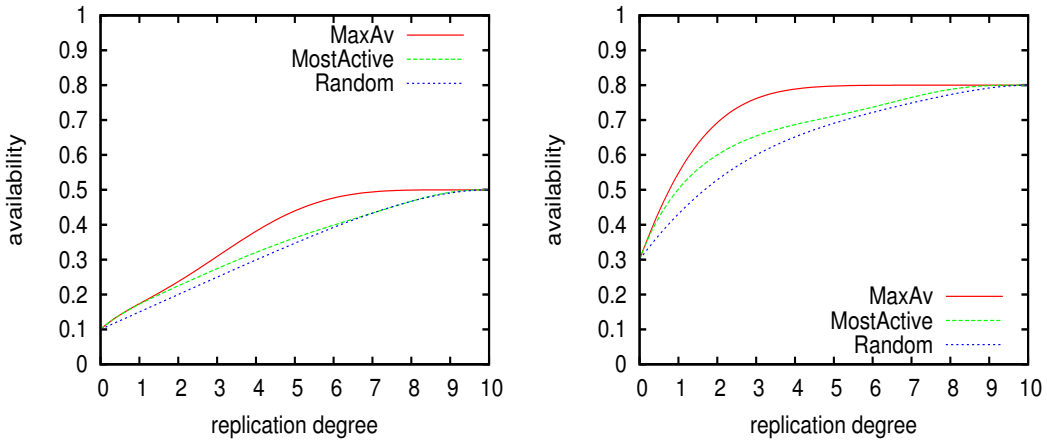
(d) FixedLength (8hours)

Figure 5.3: Facebook-ConRep: Availability

5.4.5.2 Availability-on-Demand vs. Replication Degree

As we have seen, availability does not reach 100% even if all the friends are employed for replication. Instead, availability-on-demand-time reaches 100% with only 5 replicas (for *MaxAv* placement and *Sporadic*), as shown in Figure 5.5a, while *MostActive* and *Random* replica placements require 7 and 9 (thus employing 70% and 90% of friends).

The achievable availability-on-demand-activity is even higher than the above, as depicted in Figure 5.6 for all online time models. This result is important, as it means, for a small replication degree, a user's profile can be made highly available during friends' activity times. We also noticed higher performance of the *MostActive* replica placement approach.



(a) FixedLength (2hours) (b) FixedLength (8hours)

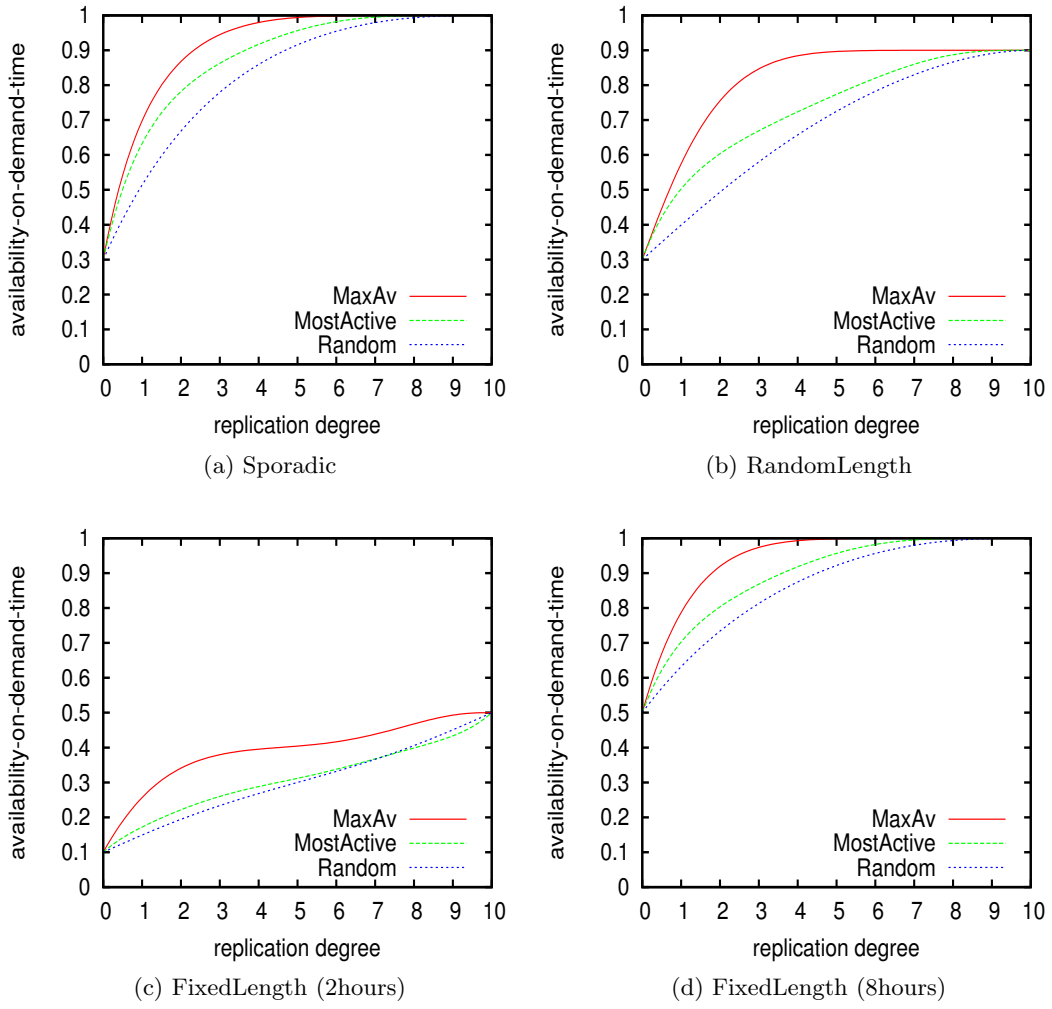
Figure 5.4: Facebook-UnconRep: Availability

5.4.5.3 Update Propagation Delay vs. Replication Degree

Non-intuitively, the update propagation delay increases with the replication degree, as depicted in Figure 5.7. However, this can be understood, as explained in Section 5.2.3.3, this metric represents the maximum delay for an update to reach all replicas; hence, increases with number of replicas, if their total non-overlapping time increases. As *MaxAv* replica placement algorithm chooses replicas with lesser overlapping times, it incurs the highest delay, as compared to the other placement approaches. The delay is lower for *Sporadic* as compared to the other online time models, since the replica nodes can contact each other more often due to their intermittent online connectivity. Note that, even though the delay seem to be unacceptably high, in general, the *observed* delay (refer Section 5.2.3.3) would be much lower. The delay is expected to be lower for the *UnconRep* case, as external communication means are used for update propagation.

5.4.5.4 Effect of the session length in *Sporadic* model

We illustrate the effect of the length of a session in *Sporadic* on the performance metrics in Figure 5.8 for the case of Facebook. The session length is shown in log scale. We considered a fixed replication degree of 3 as we observed a flattened performance for higher replication degrees (see Figure 5.3). The plots (in Figure 5.8) affirm that an increased session length can boost the performance. For a session length above 10^4 sec, even the achieved availability reaches 100%. The propagation delay significantly decreases with the session length as shown. A higher replication degree (> 3) will ensure same availability even with lesser session length which is not discussed here.



(a) Sporadic

(b) RandomLength

(c) FixedLength (2hours)

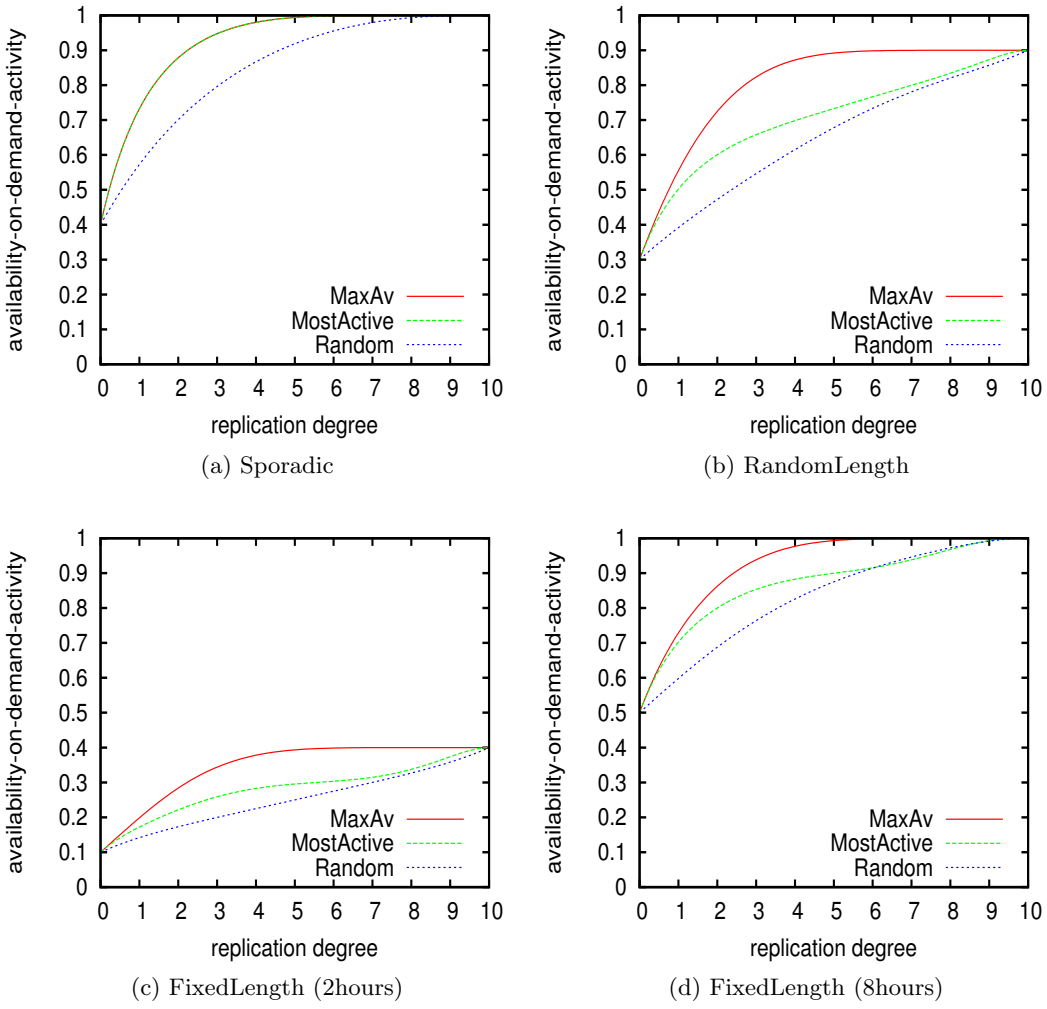
(d) FixedLength (8hours)

Figure 5.5: Facebook-ConRep: Availability-on-Demand-Time

5.4.5.5 Effect of the user degree in *Sporadic* model

In Figure 5.9, we explore the performance behavior of the algorithms as the user degree (i.e., number of friends (followers) in Facebook (Twitter)) is varied. We considered all the users with a fewer number of friends (i.e., between 1 to 10) while allowing the replication degree to be the highest possible for a given user degree. The plots suggest that the availability for users with a fewer friends is lower and increases with the user degree. Yet, we observed an availability-on-demand-time/activity of 1 for all the user degrees (plots are excluded for brevity reasons).

Since all the friends are allowed to be used as replicas, all the algorithms achieved the same availability as shown in the Figure 5.9a. But the actual number of replicas used by different algorithms to achieve this availability is found to be different, which is implied by the varied propagation delays (shown in Figure 5.9b). The *MaxAv* uses lower number of replicas compared to other algorithms (and hence, a lower delay).



(a) Sporadic

(b) RandomLength

(c) FixedLength (2hours)

(d) FixedLength (8hours)

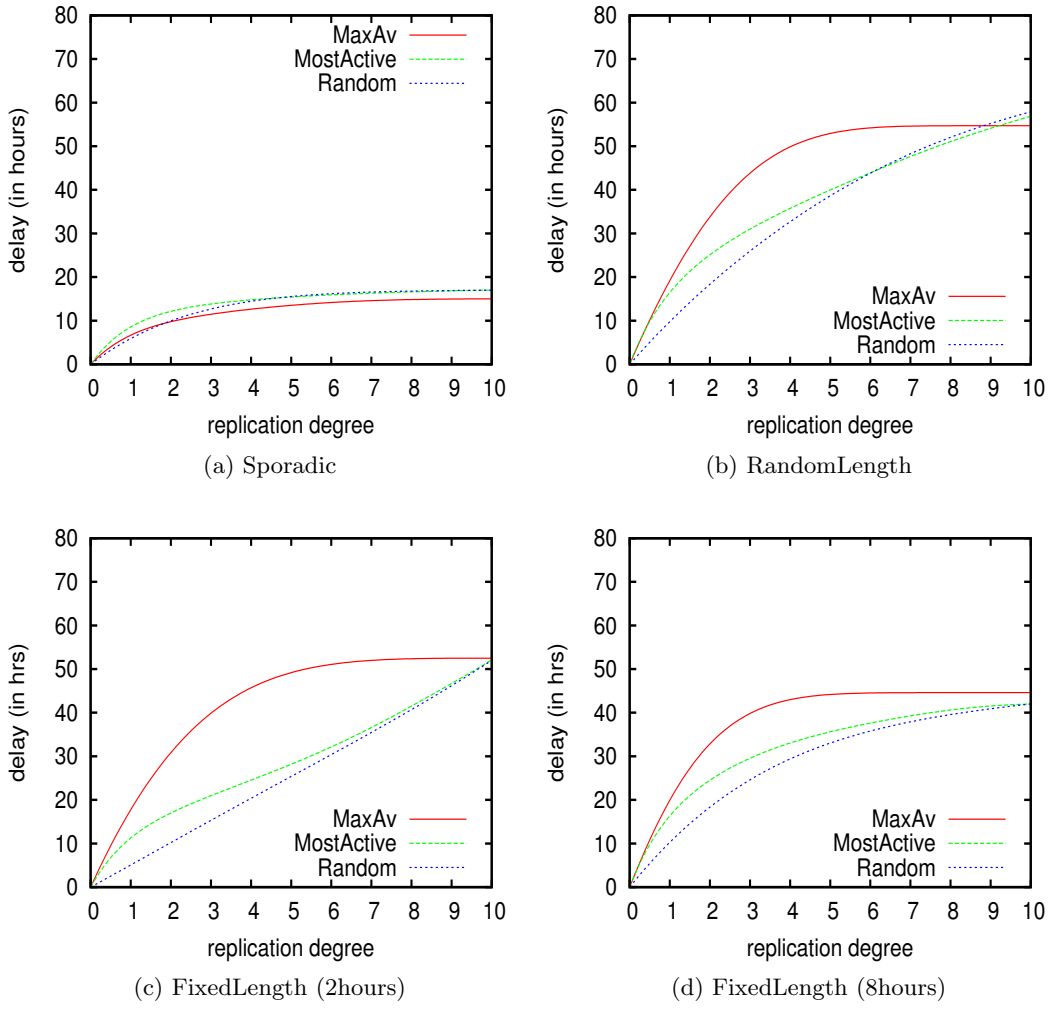
Figure 5.6: Facebook-ConRep: Availability-on-Demand-Activity

5.4.6 Twitter

We observed similar trends in the results for the Twitter dataset for all the metrics. The plots for the availability metric are presented in Figure 5.10. The availability-on-demand-time, in the case of *FixedLength* (8*hrs*) (Figure 5.11d), does not reach the maximum (like the case of Facebook Figure 5.5) because friends of some users are not at all connected to any of the users replicas, but still considered for the metric computation. Such disconnected friends can access the corresponding users profile if and only if the user extends his online time.

5.4.7 Discussion

Availability is a critical concern for decentralized OSN infrastructures. From the empirical evaluations above, we justify the feasibility of decentralized online social networks for privacy-conscious users that typically expect their profiles available only to their



(a) Sporadic

(b) RandomLength

(c) FixedLength (2hours)

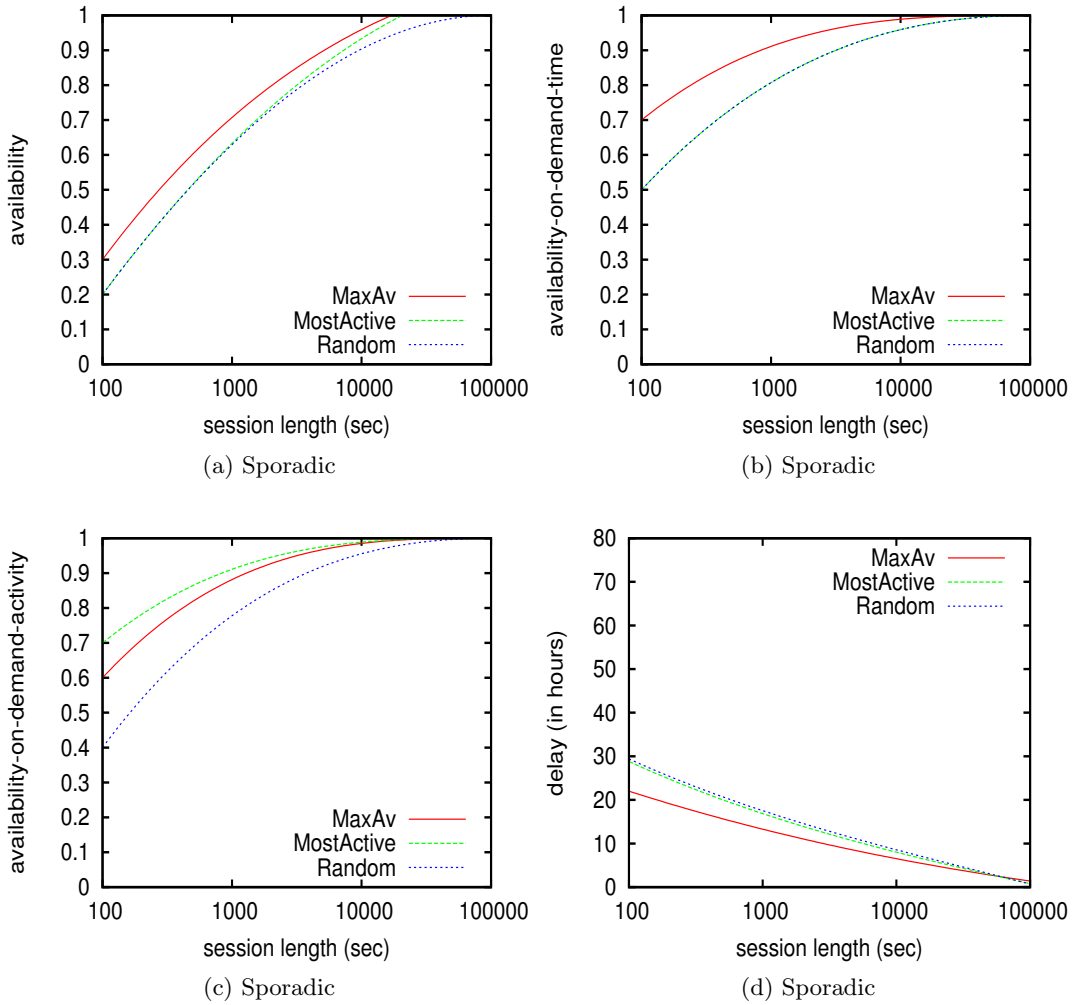
(d) FixedLength (8hours)

Figure 5.7: Facebook-ConRep: Update Propagation Delay

friends in the network (i.e., high availability-on-demand-time/activity). We observed that typically a low replication degree ($\sim$ 40%) achieves high availability-on-demand for *Sporadic*, *RandomLength* and *FixedLength* (8*hours*), i.e., for realistic online time modes in which the users are online for reasonable durations.

We highly emphasize that, ideally higher availability-on-demand-time/activity and lower availability are desirable for privacy-aware OSN design; higher availability of profile replicas can be seen as higher potential exposure (for example, from security attacks) to non-friend users and thus higher vulnerability. In the above study, we proved that decentralized OSNs using F2F-based replication are ideal candidates for this purpose.

Also, *MostActive* replica placement is a promising approach for decentralized OSNs as it is computationally simpler and does not require knowledge of the user online times, as opposed to *MaxAv*. Activities of friends and online time connectivity among them can be estimated locally based on historical data. *MostActive* also achieves a good compromise between availability-on-demand and update propagation delay.

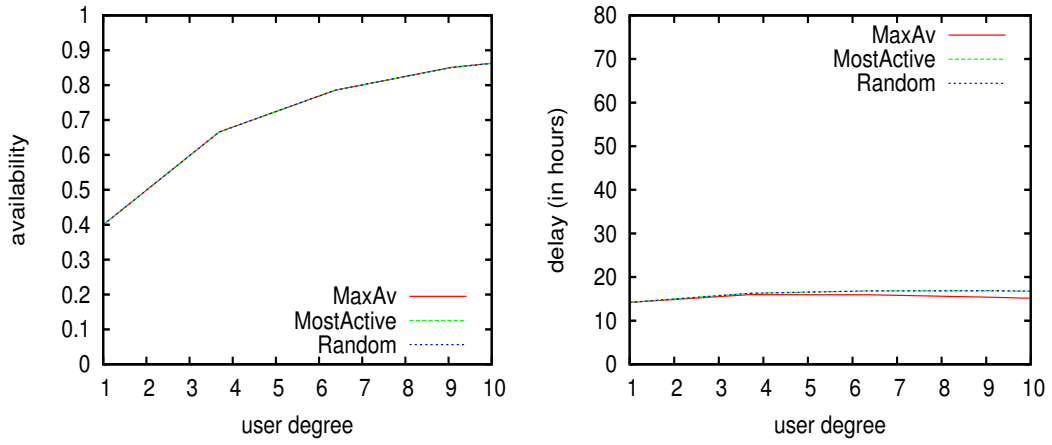


(a) Sporadic (b) Sporadic

(c) Sporadic (d) Sporadic

Figure 5.8: Facebook-ConRep: Effect of session length in *Sporadic* model (replication degree 3)

The update propagation delay seems to be a big challenge towards the realization of decentralized OSNs; we empirically found delays of ∼2 days for some online time models. Although, this is not the *observed delay* (refer Section 5.2.3.3) as perceived by the users which would be significantly lower, this delay may be still unacceptable to most users. In order to reduce the delay, the non-overlapping times among profile replicas have to be reduced; this could be achieved with longer online times of a certain core group of friends. Alternatively, the decentralized OSNs can make use of a third-party services (e.g., content distribution networks (CDN), DHT, cloud storage etc.) for exchanging updates. However, this would require encryption of the updates exchanged.



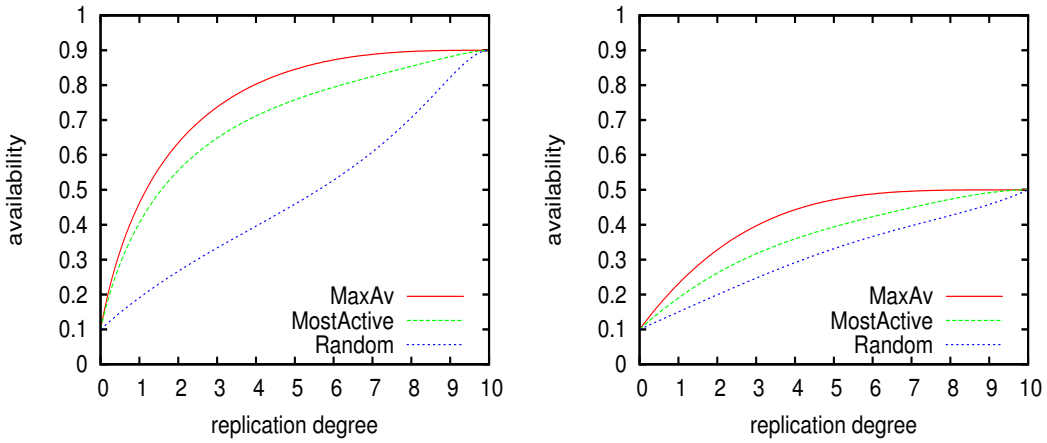
(a) Sporadic (b) Sporadic

Figure 5.9: Facebook-ConRep: Effect of session length in *Sporadic* model (replication degree 3)

5.5 Related Work

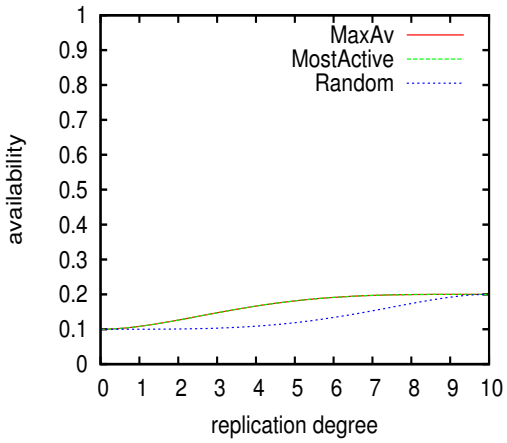
In the literature, there are many proposals on privacy-aware decentralized social networks, which were discussed in detail in Section 2.5.3.2. However, all of these works do not aim at experimental evaluation of availability or other performance metrics. In [114], a decentralized OSN is proposed, where a user's profile content is stored at his own machine called as virtual individual server (VIS). VISs self-organize into P2P overlays. Three different storage environments, namely, cloud storage, P2P storage on top of desktops, a hybrid storage were considered, and various performance issues: availability, cost, and privacy were studied. In desktop-only storage model, profiles are replicated on a user's friend nodes. However, this paper neither considers the online times of peers nor replication placement policies and their implications on the performance of the system.

The authors in [61], [115] deal with friend-to-friend storage systems and our work complements to them. They argue that F2F based storages are more desirable for some application scenarios over conventional P2P based alternatives. A major obstacle in deploying a distributed storage infrastructure in peer-to-peer systems is storing data reliably using nodes that have little incentive to remain in the system. This problem can be circumvented using storage architectures based on existing social relationships instead of randomly. This approach provides incentives for nodes to cooperate and results in a more stable system which, in turn, reduces the cost of maintaining data. The cost of this approach is- decreased flexibility and storage utilization. The work in [61] targets file storage application scenario and proposes a F2F based solution. An analytical and experimental evaluation is presented to prove the merits of F2F systems over conventional P2P systems. The reliability was approached by replication and by incentivizing peers to stay online. A more recent empirical study of availability of F2F systems is pursued in [115]. An instant messaging service dataset is used for the experimental

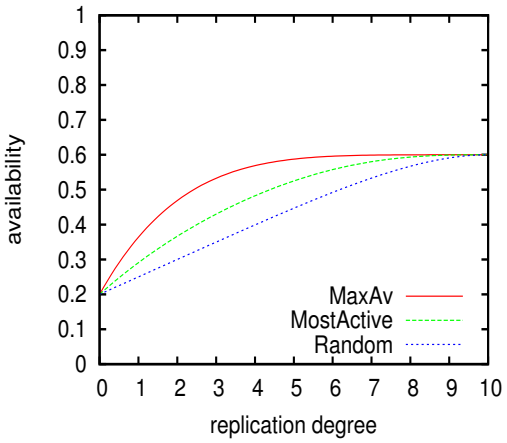


(a) Sporadic

(b) RandomLength



(c) FixedLength (2hours)



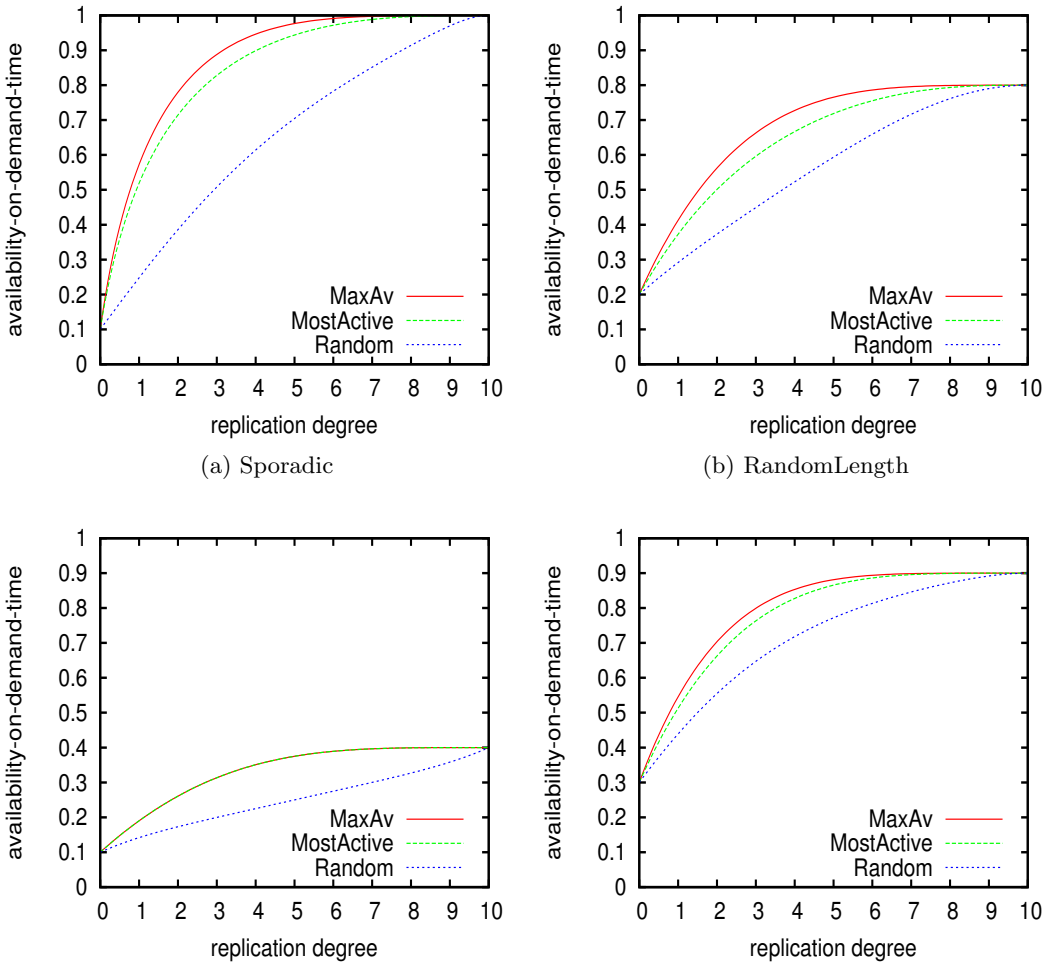
(d) FixedLength (8hours)

Figure 5.10: Twitter-ConRep: Availability

study. In our work, we study systematically and analyze various challenges for realizing the decentralized OSNs and employ data traces from two real and well-known OSN applications: Facebook and Twitter. We also consider two separate cases of connected and unconnected replicas.

5.6 Conclusion

In this chapter, we introduced the important performance metrics for decentralized OSNs, experimentally analyzed the trade-offs and derived feasibility conditions for their realization in practice. Based on the experimental evaluation and our user online time modeling, we conclude that the implementation of a decentralized friend-resident social network is feasible under certain realistic requirements on the user online times, which determine the necessary replication degree and the resulting availability of the system. In the next chapter, we will extend this feasibility study into a decentralized online so-



(a) Sporadic

(b) RandomLength

(c) FixedLength (2hours)

(d) FixedLength (8hours)

Figure 5.11: Twitter-ConRep: Availability-on-Demand-Time

cial network and introduce the *My3* system. A detailed experimental study is conducted exploiting the lessons learned in this chapter.

Chapter 6

My3: a Privacy-aware Decentralized Social Network

If you're not paying for something, you're not the customer; you're the product being sold.

Andrew Lewis

6.1 Introduction

Online social network paradigm has taken the Web 2.0 into unprecedented scales by offering innovative tools for networking among users and distributing the user-generated content. As stated in the previous chapter, the conventional social networks (e.g. Facebook.com, Google Plus) have recently seen an explosive growth. Facebook has currently nearly 900 million users active on the service at least once in a month. As a result, these OSNs have become store houses of unprecedented amount of data in the form of messages, photos, links, and personal information. Online social network portals continue to evolve towards one-stop hubs for content in a way that influences the future of the Internet [91]. However, social networking portals that are operated in a cloud infrastructure maintained by a single authority (e.g., Facebook or Google) will strategically have greater stakes in protecting the interests of the advertisers than addressing the privacy requirements of the users. During sign-up time, users consciously or unconsciously permit the organizations to share their personal information with third-parties in whatever form the organizations choose to [70]. In addition, the leakage of personal information from OSNs can be associated with the user activity on non-OSN sites as well [79]. Moreover, as social information owned by a single authority grows, so does its financial power. Even if we trust that the provider is motivated to protect user data, large-scale data breaches are still possible as reported recently. In order to address privacy concerns of OSN users, research community has resorted to the decentralized (often

P2P-based) paradigm for OSN content management. Replacing the big-brother with a community of users, enables OSN users to have more control on their profile content.

In this chapter, we present *My3*[1], a decentralized online social network that operates based on the resources contributed by its participating users. First, we outline the system architecture and later focus on the distributed storage layer. Specifically, we propose a number of profile replication algorithms that can be independently employed by users to meet different performance objectives of their choice, namely high profile availability, high data freshness (i.e., low delay for data consistency), low access delay, low storage overhead or a certain combination of them. As we demonstrate here, *My3* makes a user's profile accessible to all of his friends in the social network (and only to them) even when he, himself, is offline, by means of appropriate replication schemes. Profile updates are propagated among replicas, so that eventual consistency is met. As opposed to completely centralized approaches (e.g., Facebook) or decentralized approaches that employ always-on personalized servers [19] for realizing online social networks, *My3* represents a third (hence "3" in the name) alternative, which builds a decentralized OSN by replication schemes that exploit overlaps in online times of trusted friends.

The design of the *My3* system exploits several properties of the conventional online social networks: (1) a user's friends are clustered in a few geographical locations and almost all the content accesses are initiated from these locations (2) access patterns and times of content accesses can be approximated with high precision to a large extent. By employing real data traces from Facebook and Twitter, we experimentally prove the effectiveness of our replication algorithms towards their respective goals when jointly or independently chosen by users. According to our results, a total-not necessarily continuous-online time of 40 minutes by a user, is enough for achieving availability higher than 90% with 4-5 profile replicas.

The remainder of this chapter is organized as follows: In Section 6.2, we discuss the background behind the *My3* system design. The storage layer is discussed in Section 6.4, followed by replication algorithms. In Section 6.7, we evaluate the performance of the *My3* design w.r.t. several criteria. In Section 6.9, we discuss the related work and, finally we conclude in Section 6.10.

6.2 Motivation

In this section, we explain the properties of the conventional social networks that drive the design of *My3* and its replica selection algorithms.

We observe that every user in an OSN has friends scattered over a limited set of geographical locations (e.g., his home town, working location, home country, and location of previous institute.) as shown in Figure 6.1. This fact can be exploited to choose replicas for a user's profile to be located in one or more of these locations so that content stays in the proximity of the user's friends.

[1] *Mythri* ("My3") is a Sanskrit word which means *friendship*.

Figure 6.1: Geo-clustering of a user's friends in Facebook.

Moreover, users in an online social network exhibit certain online time patterns [44], [111] which can be exploited in choosing the replication points so that replicas' online time patterns overlap with that of friends of a user. Note that, in a typical OSN, most of the user's profile content is accessible only to his friends, unlike typical web content which is world-accessible. The *My3* system exploits the trust relationships among friends to improve the availability. We assume that a user of *My3* runs the client on his office or personal laptop/computer. Hence, we use the terms *user* and *node* interchangeably.

We consider optimizing at a single user level in all our algorithms and not system-level optimization, as each user runs these algorithms independently from others in a distributed setting. *My3* allows users to personalize their profile configuration in several dimensions such as availability, responsiveness, privacy risk, etc. We believe that a single global configuration policy for all the users in the system offered by typical conventional social networks like Facebook, deprive the users their autonomy on their own data.

6.3 System Overview

As mentioned earlier, the *My3* system exploits the trust relationships among friends and social network connections to improve the availability and search performance of the system. A user u's profile content is hosted only on a set of self-defined trusted nodes, which enforce access control on the content on behalf of the user. This set of trusted nodes for a user is referred to as his *trusted proxy set (TPS)*. The TPS members for a user are properly selected with respect to the availability and performance goals. The computation of the set TPS based on a user's social graph is explained in next section. Each user u is identified by a unique identifier denoted by UId_u. Note that a TPS is a set of UIds.

The *My3* system employs a distributed hash table (DHT) hosted at the resources contributed by the users. This DHT is used for storing meta information, e.g., the current IP address of a user's *My3* client. A user u and his TPS mapping is stored in

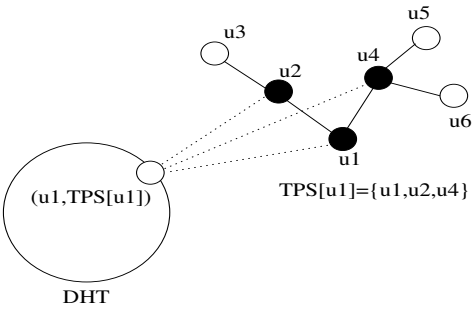


Figure 6.2: The *My3* storage layer.

the DHT in the form of *(key,value)* pair with *key* being the UId_u and *value* being the members of the TPS. Using cryptographic signatures, it should be trivial to test the authenticity of such an entry in the DHT. This user-to-TPS mapping in the DHT is useful for contacting the nodes where the profile of a particular user is stored.

We assume that, with a reasonable replication factor, one can ensure that the data items stored inside the DHT are highly available in spite of node churn. As a trusted storage is not required by the system design, such a DHT could be hosted at a highly available cloud storage or in publicly available OpenDHT-like services [104]. The *My3* storage architecture is illustrated in Figure 6.2. Therein, the user u_1 has 5 friends in the OSN, namely u_2 to u_6. The set $TPS = \{u_1, u_2, u_4\}$ is shown in the figure and a mapping between u_1 and $TPS[u_1]$ is inserted into the DHT. The user social graph is represented as *online time graph*, which is explained in the next section.

6.4 Storage Layer

The social network graph is denoted as $G(U, R)$, where U is the set of users represented by the vertices in the graph and R is the set of friendship relations represented by edges. For example, an edge between two vertices u_1 and u_2 models the fact that users u_1, u_2 are friends. We assume that friendship relationships are symmetric. This is the default assumption in current OSN applications, e.g., Orkut, Facebook. We use the notation $N_G(u)$ to represent the set of neighbors (i.e., friends on the OSN) of user u in the social graph G, and $N_G[u]$ to represent $N_G(u) \cup \{u\}$.

We assume that each user u in the social network is characterized with two parameters: his *geographical location* and *online time period*. We exploit location information of friends of a user in order to place the user's profile data as close as possible to the friends who most frequently access the data for getting profile updates. This is quantified by the metric *access cost* $C_{u_1}^{u_2}$ between two geographical locations/users/nodes u_1 and u_2, which is defined as the IP latency between those two locations, for example, a measured in the Verizon dataset [28].

Online time period represents the usual time that the user is online in the social network. This is a continuous/discrete time period with a predefined granularity (e.g.,

minutes, hours) during which the user is active on the network and contributes resources such as the network bandwidth and storage space through his OSN client. This parameter can be either a user input to the client or approximated by the client from the user's online history (for example, as done in the later part of the chapter). The user is considered to be offline beyond this time window. We denote the location and online time period parameters for a user u as L_u and OT_u respectively. Given two users u_1 and u_2's locations and online time settings, we argue that they can contact each other and thus exchange data if and only if their online time intervals overlap, which we represent by the condition that $OT_{u_1} \cap OT_{u_2} \neq \emptyset$.

6.4.1 Trusted Proxy Set

Each user u selects some of his neighbors as *trusted* nodes. The user trusts these nodes both for storing his profile content and for enforcing access control on the access requests. We believe leveraging mutual trust relationships for access control enforcement, in place of encryption-based-access control simplifies the system to a great extent, especially given that a typical user in any OSN has millions of data objects but of very small size. We envision that users mutually cooperate for hosting content and delegating access control with some social contracts, even though *My3* design does not assume that trust is mutual between friends. However, the intuition is that users do not breach the delegation responsibilities because of social pressure and monitoring. Alternative solutions, which employ encryption mechanisms for access control and content storage [50], not only involve complicated key management issues, but also, are highly inefficient in terms of storage overhead, as the same data item may need to be encrypted multiple times for different users with different access rights. However, trust in a user, may not translate to trust in his machine/ node. We acknowledge that detecting compromised nodes is a research problem in itself and assume that any of the existing mechanisms [63] can be deployed for this purpose.

Let $T(u) \subseteq N_G(u)$ be the set of trusted users/nodes for a user u. $T[u]$ also includes the user u himself in the set of trusted nodes. The user selects a subset of these trusted users for hosting his content. We call this set as *trusted proxy set (TPS)* ($TPS(u) \subseteq T(u)$). The content of u is stored on the members of the set $TPS[u]$ ($= TPS(u) \cup \{u\}$).

Next, we describe several algorithms for the computation of the set $TPS[u]$. These algorithms build an *online time graph* for each user and compute the above set from this graph.

Definition *Online time graph:* for a user u (denoted by OG_u) is defined as $(N_G[u], E)$ where $N_G[u]$ is the set of vertices and E is the set of edges, such that

$$\forall v_1, v_2 \in N_G[u], \ \exists \text{ an edge}(v_1, v_2) \in E \ \text{ iff}$$
$$(v_1 \in T[u] \vee v_2 \in T[u]) \wedge (OT_{v_1} \cap OT_{v_2} \neq \emptyset)$$

Next, we specify the following two conditions on the graph OG_u, which are necessary and sufficient in order to compute a valid storage configuration.

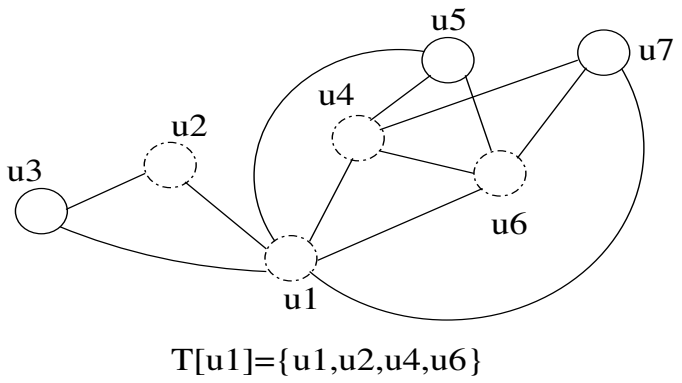


Figure 6.3: The *online time graph* for a user u_1 (OG_{u_1}).

1. OG_u must be connected. Only then, every user in the set $N_G[u]$ can access u's content.

2. The sub-graph induced by the set $T[u]$ i.e., the graph $OG_u[T[u]]$ must also be connected, in order to allow content synchronization across TPS members pass through only trusted nodes[2].

We suppose that each user constructs OG_u offline locally from online time (OT) specifications of his friends. The construction of OG_u is explained with the following example. Assume a user u_1 with neighbors in the OSN u_2 to u_7 and their locations set as follows: L_{u_1} is *Switzerland*, L_{u_2} and L_{u_3} are *India*, and finally the rest are *US-West*. Assume OT set to 8am to 5pm local time for all users. Let $T[u_1] = \{u_1, u_2, u_4, u_6\}$. The resulting OG_{u_1} is shown in Figure 6.3.

We found that, in the case of real-world datasets (explained in Section 6.7, such a graph $OG_u[T[u]]$ is connected for a realistic online time model for more than 95% of the users.

6.5 Replication Strategies

In this section, we describe several algorithms used by *My3* for replica selection, based on the lessons learnt in the empirical evaluation done in Chapter 5. Each algorithm computes the TPS optimizing a certain objective function, as explained below. The *My3* system adopts the *ConRep* approach introduced in Chapter 5 which mandates that replica time connectivity graph should be connected.

As social relations evolve, there will be updates in a user's social graph. Moreover, breach of trust or the social contract to enforce access control by his friends may result in updates in the set TPS. We believe that mutual social contracts (i.e., reciprocative hosting of data between users) restrict users from maliciously exploiting their hosted data after their removal from the TPS. Once a node v is removed from a user u's TPS,

[2]However, as long as the first condition is met, nodes from the set $T[u]$ can be removed one by one until the resulting induced graph becomes connected.

it is no longer contacted for u's content by his friends. We assume the user periodically invokes TPS computation process to accommodate the updates made on OG graph because of updates in the set $T(u)$ or updates in friendship relationships. When there is a change in the location of some trusted nodes, the graph OG_u may get disconnected. Noticing this, the user u should adjust his online time frame OT_u, if required, in order to keep his profile available to changed online times of such friends.

6.5.1 Maximizing the availability *(MaxAv)*

In this approach, the trusted friends that maximize the availability of the user profile are chosen as replica locations. The maximum achievable availability for a user u is limited by $\mid \cup_{f \in T[u]} OT_f \mid$. Hence, the replica selection algorithm should choose the minimum number of replicas/friends that jointly achieve this availability. We model this problem as the conventional *set cover problem* with the set to be covered (the *universe*) chosen as $\cup_{f \in T[u]} OT_f$. Since finding an optimal solution for the set cover problem is NP-hard, we solve the problem with a greedy algorithm as follows: Initially, the node u is added to the TPS. At each step, a trusted friend that has the longest non-overlapping (i.e., uncovered) online time as compared to the current TPS members is added to the $TPS[u]$ until no improvement is observed in the achieved availability. Only the friends that are connected in the online time graph to any of the current $TPS[u]$ members are considered in each step. The availability of TPS is the fraction of sum of its member's online times over a day (i.e., 24 hours).

6.5.2 Minimize the number of replicas *(MNR)*

The MNR approach minimizes the number of replicas to be maintained for a user, so as to minimize the storage and replica management overhead. This approach exploits the fact that the set TPS can be modeled as the *minimum connected dominating set (MCDS)* on the graph OG_u, with the additional constraint that the members of the MCDS must belong to $T[u]$. Hereby, we modify a greedy algorithm from [107] to solve this variant of the MCDS problem.

In graph theory, a *dominating set* for a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is joined to at least one member of D by some edge. Different possible dominating sets for a graph are shown in Figure 6.4. A *connected dominating set* of a graph G is a set D of vertices with two properties:

1. Any node in D can reach any other node in D by a path that stays entirely within D. That is, D induces a connected subgraph of G.

2. Every vertex in G either belongs to D or is adjacent to a vertex in D. That is, D is a dominating set of G.

A *minimum connected dominating set* of a graph G is a connecting dominating set with the smallest possible cardinality among all connected dominating sets of G. In Figure

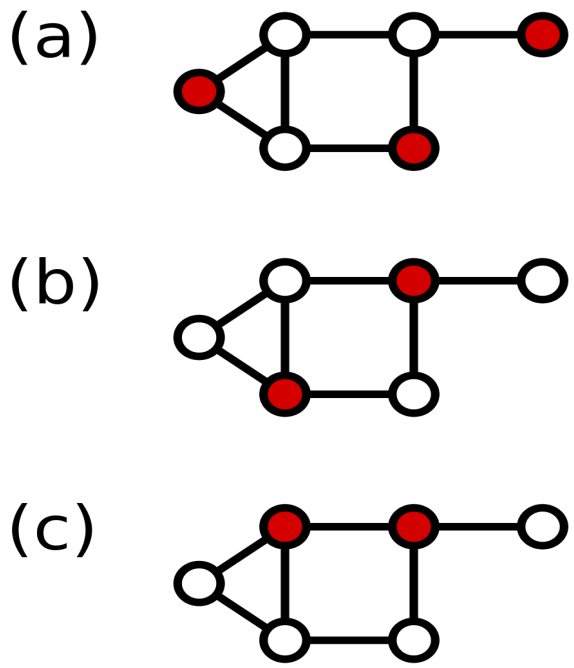


Figure 6.4: Dominating sets for a graph (vertices in dark/red) [1].

6.4, a dominating set, a minimum dominating set, and a minimum connected dominating set were shown in that order (set of vertices in dark/red).

Algorithm 6.1: The MNR algorithm.

1: Mark all $v \in OG_u$ as *white*
2: Mark u as *black*
3: Mark all neighbors of u in OG_u as *grey*
4: **while** $\exists$ a *white* node in OG_u **do**
5: Select a *grey* $v' \in T(u)$ such that v' has the highest number of *white* neighbors in OG_u
6: Mark v' as *black* and its neighbors as *grey*
7: **end while**
8: $TPS[u]$ is the set of all *black* nodes in OG_u

6.5.3 Minimizing the update propagation delay *(MPD)*

This *update propagation delay* among replicas is the delay in time between the time instance, an update occurs on a user profile at one of the replicas and the instance and the update reaches all the other replicas. We explained the calculation of this delay in Chapter 5 in Section 5.2.3.3.

This algorithm minimizes the worst case propagation delay for the user u. This algorithm works as follows: we compute a modified weighted online time graph OG''

from OG by assigning weights to edges between trusted nodes, equal to the propagation delay between the corresponding end nodes. Then the problem of the TPS computation is finding the $MCDS$ in graph OG'' such that the weight of the longest shortest path between any two nodes in TPS in OG'' is minimum. The $WP(S)$ function returns the weight of the longest shortest path in the sub-graph induced by the set S on OG''.

Algorithm 6.2: The MPD algorithm.

1: Mark all $v \in OG''_u$ as *white*
2: Mark u as *black*
3: add u to TPS
4: Mark all neighbors of u in OG''_u as *grey*
5: **while** $\exists$ a *white* node in OG''_u **do**
6: Select a *grey* $v' \in T(u)$ such that $WP(TPS \cup v')$ is the minimum
7: Mark v' as *black* and its neighbors as *grey*
8: add v' to TPS
9: **end while**

6.5.4 Minimizing the access cost *(MAC)*

The MAC approach prioritizes only the access cost for each friend in a user's social network. Hence, for every user v in OG_u, it chooses the nearest (i.e., with minimum access cost) trusted node into the set $TPS[u]$. This algorithm always chooses all the trusted nodes into the TPS set, as for each trusted node the nearest node is itself. Thus, it uses all the possible replicas which results in highest possible replication i.e., equal to the number of trusted friends.

6.5.5 Maximizing the replication gain

This approach quantifies the *replication gain* of a given subset of trusted nodes set (x) and, explores the entire solution space to pick the right set with the minimum effective cost as TPS. The storage cost is measured in terms of the total cost incurred for accessing and updating the profile content by friends, in addition to that of replica synchronization among all TPS members.

Replicating a user's profile increases the availability of the profile but reduces the average access cost per friend in accessing the profile. In addition, it induces an overhead in the form of update propagation delay among replicas. A typical user of a decentralized OSN needs a flexible framework to fine tune each of these parameters. To this end, all these three factors are merged in *My3* into a weighted objective function with tunable weights to each of the factors.

The algorithm works as follows: the TPS (i.e., set x in eq. (6.1)) is initialized to u. Then one member (i) at a time is added to the set, from the set of trusted friends,

that maximizes the following objective function, until the resulting TPS is a minimum connected dominating set over the graph OG_u.

$$\Big[w_1 \cdot \frac{availability(x \cup \{i\}) - availability(x)}{availability(x)} + w_2 \cdot \Big|\frac{avg(\{C_v^x\})}{avg(\{C_v^{x\cup\{i\}}\}) - avg(\{C_v^x\})}\Big| - w_3 \cdot \frac{WP(x \cup \{i\}) - WP(x)}{WP(x)}\Big] \text{ where- } v \in N_G(u) \tag{6.1}$$

The access cost between a node (v) and a set of nodes (x) is the access cost between v and its nearest node in x. The function WP is explained above in the *MPD* algorithm. This algorithm is referred to as *Hybrid* in the evaluation (Section 6.7).

6.6 Data Consistency

As different replicas of the profile accept update requests from the friends of the user in an asynchronous way, there is a need for synchronizing the profiles on all replicas. We propose that after every update, the concerned replica *pushes* the update to other TPS members during their online time frame. Note that $OG_u[T[u]]$ is connected. Assume that each TPS member is informed of other members by the user u during TPS creation. We adopt *eventual consistency* for the *My3* system. Until recent updates reach a replica, it continues to serve access requests with out-dated content, which is acceptable in an eventually consistent system.

My3 views the content of a profile as a collection of data objects e.g., a status message, user's metadata, a photo album, a photo. A data object (say, a photo album) can be further decomposed into another collection of objects (resp. photos). *My3* employs vector clocks of size the number of TPS members in the system. It maintains one vector clock per object. A vector clock of an object is updated when there is an update in the object. However, they are not updated when its constituting objects' vector clocks are updated. For example, when a comment is posted on a photo, thus the photo object is updated, the corresponding vector clock of the photo is updated and that of the photo album object is kept intact. If the photo is deleted by the owner user, then the vector clock of the album object is updated as this deletion is an update on the album.

Updates on a profile are *pushed* immediately by a replica to all other replicas. In addition, when a replica comes online, it announces itself to all other online replicas and *pulls* any buffered updates, as explained below.

Each replica buffers a transaction record of an update on its copy of a user's profile until a time period (e.g., twice the maximum propagation delay for the user). This record holds all the meta information corresponding to the update so that other replica, on receiving the record, can reproduce the update exactly on the version of the replica it

hosts. Two replicas when come in contact, exchange all the records in the buffer and apply the records on the corresponding data objects. When concurrent events are detected on an object (using the vector clocks stored in the records), the two replicas have to decide on ordering the events. Even though any ordering of these events results in a valid profile only (as typical updates on an object will be append-only updates), we propose the events to be ordered according to the replica identifiers in order to achieve a total order of events among all the replicas. This results in a consistent view for the users when they access the profiles across replicas. In order to achieve this, the transaction record should contain the replica id of the replica which originally received that corresponding update event as a replica may receive a transaction record from multiple replicas because of asynchronous update propagation and nodes' intermittent online connectivity. Hence, a sample transaction record looks like the following: `(replicaId, friendId, objectId, operation, content)`. The `replicaId` points to the initial replica which received the update on a user's profile performed by a friend in his social graph identified by `friendId`. The update is performed on a data object in the profile represented by `objectId` and the corresponding update operation (i.e, a "like" or a "comment" on a photo) and the content are also present in the record.

However, this ordering resolution does not take the actual semantics of the updates into account and hence may lead to semantically incorrect profile objects occasionally. We expect the owner of the profile to inspect his profile updates time to time and fix such semantic incorrectness, though we believe that such an intervention is needed very rarely. The resulting ordering of the events must be *forced* onto other replicas which should replace corresponding object parts with the one received from the owner. Thus, the owner replica can be said to be the *leader* of all other replicas.

6.7 Evaluation

In this section, we illustrate the performance evaluation of the proposed *My3* storage in detail, using real-world datasets of Facebook and Twitter social networks. First, we present description of the datasets and then layout the criterion for evaluating the performance by introducing the metrics, followed by analysis of the results.

6.7.1 Datasets

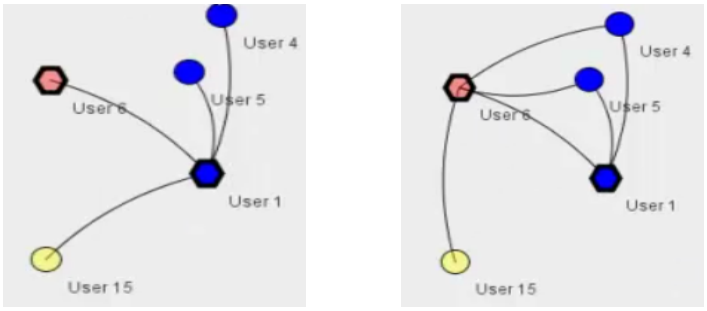
In order to model the essential parameter of *My3* algorithms, the online times of users in a social network, we needed, apart from the social network graph, a dataset with users' usual activities on the social network including the timing of the activities information [98] and their geographical locations. Two datasets- a Facebook dataset [124] and a Twitter dataset [64] met our requirements. These datasets are described in detail in Chapter 5 (Section 5.4.1).

We filtered out the users with a very little activity (less than 10 wall-posts or tweets) from the above datasets. We ended up with a total number of 13884 users for Facebook,

with the average degree 41 (i.e., friends) and an average number of 50 activities per user. For Twitter, the filtered dataset contains 14,933 users with average degree of 76 (i.e. followers).

6.7.2 *My3* simulator

We built a Java-based simulator for evaluating and demonstrating the *My3* system, which we briefly explain here with a sample user and his social network graph shown in Figure 6.5a. A user (*User 1*) shown has 4 friends in his social network. Each user is



(a) *User 1*'s social network graph

(b) *User 1*'s online time graph

Figure 6.5: Visualization of a user's social network and online time graph in *My3* simulator. Colors represent the time zones and diamond shaped nodes identify trusted friends. The red node is connected in time to both the blue and the yellow nodes where as the blue and the yellow ones do not have any overlaps.

colored based on his online time window. All the users represented by the same colored nodes have identical online times. In this example, the red and the blue, the red and the yellow online time windows overlap in time. The blue and the yellow nodes do not have any overlapped time windows and hence can exchange update only through a red node. *User 1*'s online time graph is shown in Figure 6.5b with his trusted friends shown as diamond shaped nodes (i.e., *User 6*). Figure 6.6 illustrates the propagation of profile updates among replicas and copies of the profile on replicas eventually achieving consistency. In this work flow, the user *User 1* updates his profile which is seen by his friend *User 4* when he accesses the replica stored on the user himself. *User 4* writes to the profile data. Later, the user *User 6* comes online and sees the updates previously performed by both of the above users. This was possible because of the replica on *User 6* synchronized the *User 1*'s profile replica stored on itself with the replica hosted on *User 1*.

6.7.3 Experimental methodology

All the users in the datasets mentioned in Section 6.7.1 are modeled in the above simulator and the activity stream present in the datasets is replayed among the user objects

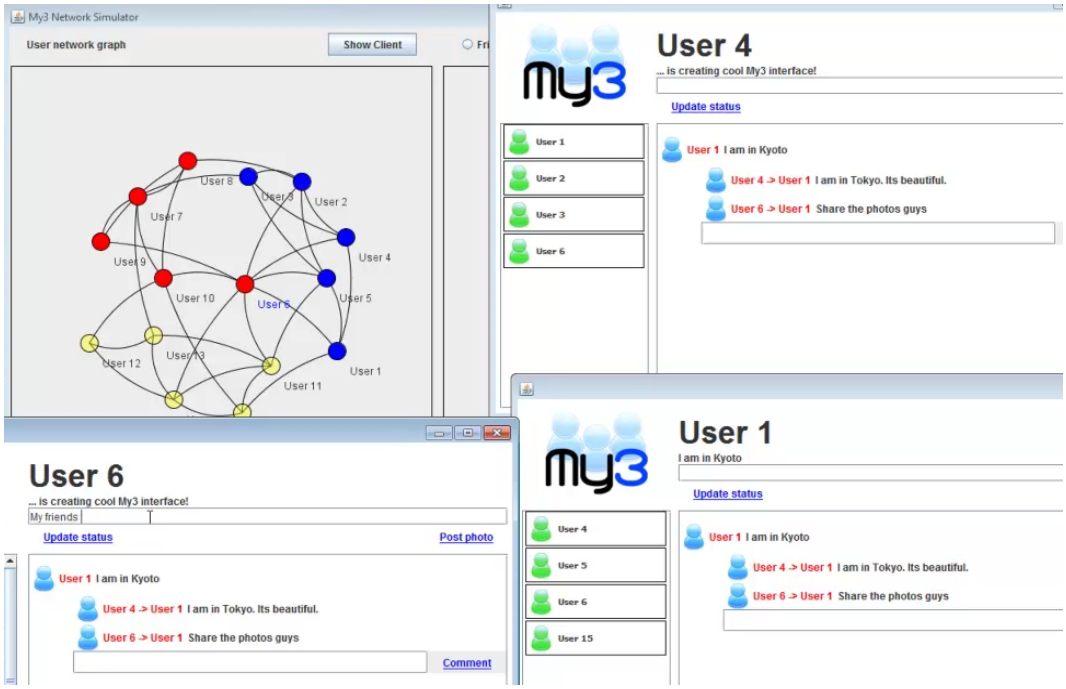


Figure 6.6: Illustration of updates propagation and *eventual consistency* in *My3*.

created, separately for Facebook and Twitter. The proposed performance metrics are quantified at the end.

However, the *My3* algorithms need two important inputs: one, the online times of users in the network and the second, the trusted friends to be used as replicas for a user's profile. In addition, the communication network latencies between two users are needed to measure the access costs in accessing a user's profile from replicas by his friends. We model these inputs as follows:

6.7.3.1 Modeling user online times

In order to approximate typical users' online durations, two possibilities exist for the context of *My3*, a decentralized social network. One possibility is to position *My3* as an alternative to the conventional social networks like Facebook and emulate the online behaviors of the users on Facebook [111] or Orkut [44] in the *My3* system for the evaluation. Alternatively, position the *My3* clients as analogous to P2P clients for communication on the Internet, like Skype and emulate typical Skype nodes' session times [69] to approximate the *My3* client's online times.

In this chapter, we model those online times as follows: from the study of user session times in Facebook, we derive an online time distribution for *My3* clients. During the beginning of the simulation, we choose an online time for a user from this distribution and the user runs his client for this amount of time, every time he is active on the social network. From this online time distribution, we plotted percentage of users in the system having a particular online time for both Facebook and Twitter cases in Figure 6.7.

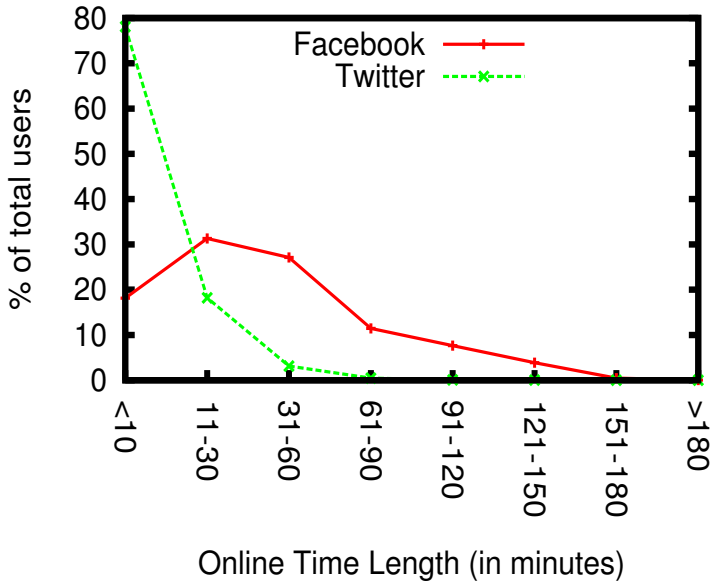


Figure 6.7: Distribution of derived user online times.

Once the length of a user's online times is chosen like above, the actual online times (time-of-the-day) are computed as follows: for each activity present in the input activity dataset, the user is assumed to be online for the above duration with the activity occurrence positioned at a random instance in this online window. This is done using the time-stamp information present with each activity in the datasets. For example, assume that the online time length chosen for a user is 5 minutes. If the user's activity is found in the dataset to be done at 8:03am, the user is said to be active on the social network from 8:00am to 8:05am if the activity is positioned at 3rd minute of the session. Likewise, a user is online during the day for a total time equivalent to number of activities in the dataset times the length of the online time associated with the user.

For the case of Facebook dataset, *My3* clients are online on average for 42 minutes with a minimum and a maximum online time of 2 minutes and 194 minutes respectively. Corresponding online times for the Twitter dataset case are: average- 9min, min-2min, and max- 156min. From Figure 6.7, one can note that for the case of Twitter, 80% (and for Facebook, 20%) of the users are online in a day for a total of less than 10 minutes. All the users in Twitter are found to be online less than 180 minutes per day where as in Facebook less than 90 minutes. 30% of Facebook users are online for 10 to 30 minutes in a day.

6.7.3.2 Selecting trusted friends

We imagine a use case for *My3* where a user manually feeds the set of trusted friends to the *My3* algorithms. However, for the evaluation sake, we model the trusted friends as the most active friends, friends who made majority of the activity on a user's profile: wallposts in the case of Facebook and tweets in the case of Twitter. We argue that this

is a natural choice as friends with very close acquaintance usually interact with a user the most, thus enjoy high degree of trust. In our evaluation, we choose *top-k* most active friends as the trusted friends with k varied from 0 to 10.

6.7.3.3 Modeling access latencies

We model the network latencies between two *My3* clients as the network latency between the corresponding geographic locations of the users as given in the input datasets. For Facebook dataset, all users are based in a single location and thus network latency between any two users is the same and set to 1 in our experiments. For the case of Twitter, we queried Twitter APIs to retrieve the locations of the users appearing in the dataset. For actual network latency statistics, we used Verizon [28] network latency dataset for the month of April 2012. If a user's (in Twitter dataset) location is not found in the latency dataset, we chose a random location from the dataset as his location. The problem of how latencies can be approximated between two geo locations is beyond the scope of the chapter and the Verizon dataset is used for exemplary purpose only. The *My3* performance trends, as such, are in general applicable to any other latency computation techniques [76].

6.7.4 Performance metrics

We enumerate several metrics to evaluate the *My3* system [98].

6.7.4.1 *Availability*

As explained in Section 5.2.3.1, it is the fraction of time in a day, a user's profile is reachable through his replicas. For example, if a user's profile is available for 12 hours in a day, the availability is 50%. Note that availability of a user's profile in *My3* is limited to the union of online times of all of his trusted friends.

6.7.4.2 *Availability-on-Demand*

This metric measures to what extent a user's profile is accessible to only his friends (in contrast to the availability of a user described above), which was introduced in Section 5.2.3.2. It is the fraction of total time, a user's friends are online (which is size of the union of their online times), his profile is available through his replicas. In a privacy-conscious social network, a user's profile is typically accessible only to his friends and hence higher availability-of-demand (even with a lower availability) is desirable.

6.7.4.3 *Propagation Delay*

It is the delay in time between the time instance an update occurs on a user profile at one of the replicas and the instance where the update reaches the last replica. The calculation of delay for a user is detailed in Section 5.2.3.3.

6.7.4.4 *Access Cost*

It is the average communication network latency between friends of a user and their nearest replica. This metric quantifies how far the replicas are located w.r.t the points of access.

6.7.4.5 *Load*

The load of a given user is the number of profiles stored on the user as part of whole social network level replication. A popular and typically most active user in the social network may end up being the trusted friend for a very high number of users in the network, thus hosting all of their profiles. A good storage algorithm should balance the load on the replicas in order to ensure fairness and minimize the maximum load in the system.

6.7.5 Results

We explore two scenarios where,

- a single system-level choice on replica placement is made and all the users in the system run the same corresponding algorithm.
- each user chooses locally the most preferred replica placement strategy that meets his criterion.

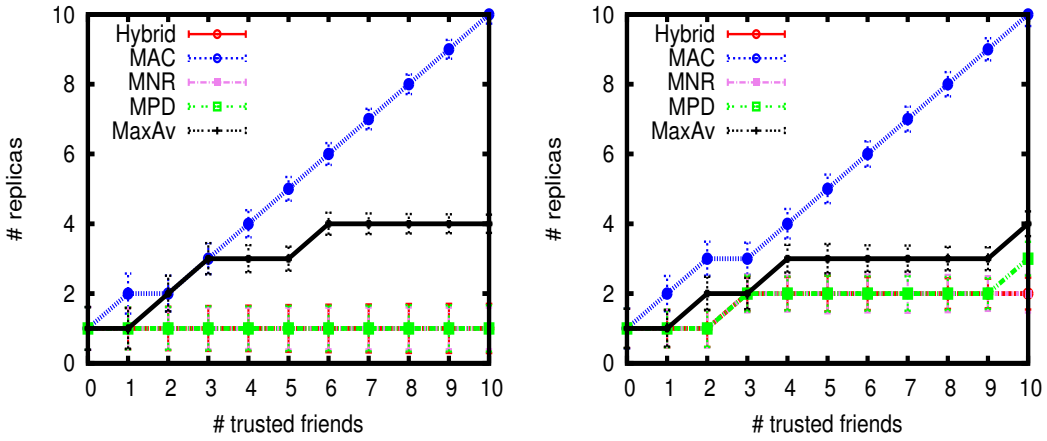
In the *Hybrid* algorithm case, all the factors in the object function were given equal weight (i.e., $w_1 = w_2 = w_3$).

6.7.5.1 System level replica placement choices

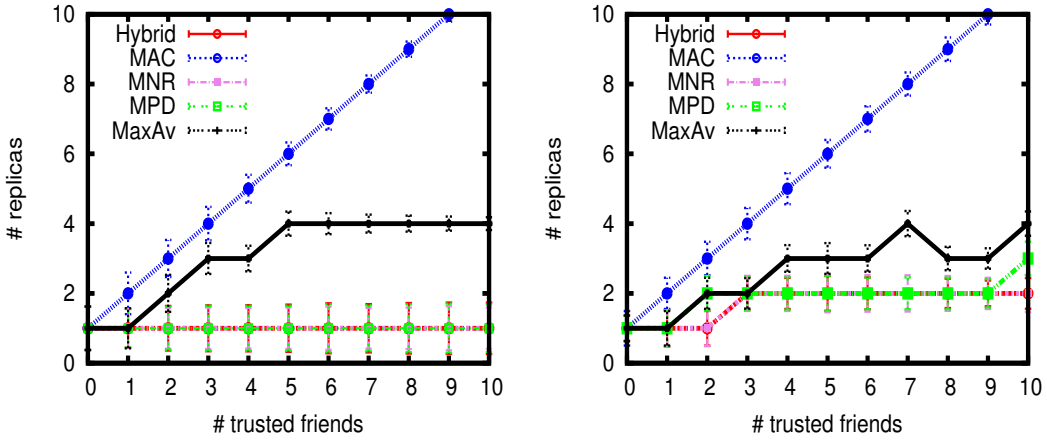
We present the observed results- first, considering all the users in the system and then, considering users with a particular number of friends (i.e., degree). We considered a degree of 20 for this case.

As mentioned earlier, the number of trusted friends is varied from 0 to 10. However, based on the objective of individual replica selection algorithms, not all the trusted friends might be used for replication. The actual number of replicas thus, chosen is presented in Figure 6.8a and 6.8b for the case of Facebook and Twitter respectively (the number of replicas counts the replica on the user client also). The system level averages are shown. As expected, the MAC algorithm uses the highest number of replicas out of all the allowed number replicas (i.e., number of trusted friends). It grows linearly with the number of trusted friends. Since some users may have a lesser degree than the maximum allowed number of trusted friends, the number of replicas is lesser for MAC case than the maximum (however, for the case of users with degree 20 (Figure 6.8c and Figure 6.8d), we can see that number of replicas chosen is same as that of number of trusted friends counting the user himself in addition). All the other algorithms show a flattened

behavior after a point as no improvement in their objective criterion is observed with increase in number of replicas. In Figure 6.9, the performance of the replica selection



(a) Facebook (b) Twitter



(c) Facebook (d) Twitter

Figure 6.8: Number of replicas chosen: (a),(b): average of all users, (c),(d): average of users with degree 20.

algorithm w.r.t the metrics described in 6.7.4 is shown. The metrics for quantified for each user in the system and average of all users is plotted in Figure 6.9a to Figure 6.9c for the Facebook case and in Figure 6.9d to Figure 6.9f. It is to be noted that for each point in the plots, the corresponding values for the metrics the achieved performance with the corresponding number of replicas shown in Figure 6.8a and Figure 6.8b. The MAC and $MaxAv$ achieve the highest availability (Figure 6.9a). But $MaxAv$ achieves the same availability as MAC with much lesser replication degree, for example at half of the replication for $k = 10$ as evident from Figure 6.8a. It is impressive to note that an availability of 90% is achieved in spite of a very less total online time of users, an average of 40 minutes from Figure 6.7 for the Facebook case and mere 9 minutes for the Twitter case. Similar availability performance of the MNR and MPD algorithms can

not be explained alone with the fact that both choose the same number of replicas as shown in Figure 6.8a. Because they exhibit differently in the case of other metrics. It is due to the chosen replicas are together online for same time window. However, the *Hybrid* algorithm makes a better selection of replicas and results in better availability with reduced delay (Figure 6.9c).

The average availability-on-demand reaches 1 for $k = 6$. Given the average degree of 41 for Facebook and 76 for Twitter, a replication degree of 6 is very promising. Note that in a privacy-friendly OSN, the profile content should be more available to friends of a user only and the actual availability is of secondary importance. The update propagation delay (denoted as "delay" in the plots) performance is depicted in Figure 6.9c and Figure 6.9f. Thanks to the highest number of replicas chosen, the MAC incurs the worst update propagation delay among replicas due to incomplete overlap among online time windows of the replicas. Even though the MPD algorithm shows up the least delay which is inline with its objective. Note that MPD chooses same number of replicas as MNR, but different ones that minimize the delay. In case of Twitter also, the MPD shows the best delay performance (Figure 6.9f). The access cost performance for Twitter case is presented in Figure 6.10. Note that for the Facebook case, all the users are from a single location (NewOrleans) and hence, this study is not applicable. As the number of replicas grows in MAC, the average access cost is significantly reduced because increased number of replicas place the content in the close proximity of many friends. This decreasing trend can be observed with other algorithms as well. $MaxAv$ stands as the next best due to higher number of replicas chosen compared to other algorithms (from Figure 6.8a).

Given the system-level averages, the actual distribution of different values of the metrics is illustrated in Figure 6.11 for the case of availability and load. We chose the case of 10 trusted friends for these plots. For around 75% users in Facebook, the availability touches 1 for $MaxAv$ and MAC algorithms. For the other algorithms, the availability is uniformly distributed. Regarding the load metric, MAC typically increases the load on users in the network because of extensive replication. There are around 0.7% of users in the network hosting more than 100 profiles in Facebook case. In case of Twitter too, we can see that MAC increases the load significantly compared to other algorithms.

In Figure 6.12, we consider only the users for a particular degree of 20 to see the trends in the performance for such users. They exhibit trends comparable to the system-at-large as shown in Figure 6.9.

6.7.5.2 User level replica placement choices

Since *My3* centers the storage design around a single user, a user in the network can locally decide the replica selection criterion based on his objectives. To this end, we studied system level performance in case of informed personal choices users make in the network. The distribution of availability (for $k = 10$) is depicted in Figure 6.13. Other metrics are skipped for brevity. Users choose one of the listed storage selection algorithms

with a uniform probability. It is interesting to note that users retain their performance benefits of a particular replication choice, even individual users in the system choose different algorithms locally. The trends observed in Figure 6.13 match exactly the ones observed for the case of single system level algorithm choice in Figure 6.11a and Figure 6.11b. The total load in the network is observed to be less than 25 per each user which is several times improvement over system-level choice (for example MAC which has a maximum load of > 100).

6.7.6 Discussion

The MAC and $MaxAv$ improve the availability at cost of increased replication. An extremely privacy-conscious user prefers to have as lesser replication as possible and thus may choose MNR for replica selection. The availability can be increased in such case only by increasing the span of online window of the user himself or his trusted friends. The $Hybrid$ algorithm should be preferred in general as it exhibits increased availability over MPD still showing similar performance for other metrics. In addition, its tunability of different factors in the objective function offers additional flexibility in replica selection. The update propagation delay can be nullified by suitable 3rd party infrastructure (like a cloud or a DHT) for update propagation. As a future study, we plan to address this problem and build such an infrastructure. In the case of highly dynamic unpredictable user online behaviors, such a storage can be used to store profile content, in addition to update propagation among replicas.

6.8 *My3* Prototype

We built a prototype of the *My3* system in Java and deployed over Planetlab [22] as a proof-of-concept for demonstration and experimental study purposes. The *MNR*, *MPD*, *MAC*, and *MaxAv* algorithms are implemented. The eventual consistency mechanism described in Section 6.6 is also implemented. For the sake of experimental study, we built a *Central Controller* which invokes various events on remote *My3* clients deployed on Planetlab nodes. For example, *addcomment* is a command which instructs one client to add a comment on one of his friend's profiles mentioned in the command arguments. This controller runs on one of the Planetlab nodes. It is also used to collect performance statistics from the clients. At the end of an experiment, the clients report various metrics to the controller. A *Central Indexer* is implemented in the place of the DHT (shown in Figure 6.2) which acts as a discovery service for the *My3* clients to discover each other. Figure 6.14a illustrates the user interface of the *My3* prototype.

We deployed 15 *My3* clients on equal number of Planetlab nodes scattered across Europe (Norway, UK, Spain, France and Switzerland) and Israel, which is shown in Figure 6.14b. All client's online times are statically configured to be 8am to 5pm in their local time zone. The central controller is installed in another planetlab node different from the above 15. The controller runs an experimental scenario and issues relevant

commands to the appropriate remote *My3* clients. We collected several performance metrics: *availability*, *availability-on-demand*, *access cost*, and *message overhead*. We report the *access cost* and *message overhead* in this thesis in Figure 6.15. The *access cost* is the average network latency measured as the average round-trip-time (RTT) using Ping requests. *Access cost* for a user is the average RTT between clients of his friends and the nearest replicas. The *message overhead* is sum of sizes of all the messages exchanged as part of update propagation among replicas. The size of a message is number of bytes the message object occupies after it is serialized by Java.

Each user chooses his own number of trusted friends. Members of the trusted set are selected randomly. The mean number of replicas (in addition to the user himself) chosen for *MNR* and *MPD* algorithms is found to be 1 and for *MAC* and *MaxAv* is 2. Figure 6.15a illustrates the access cost performance of the algorithms. Since *MNR* chooses lesser number of replicas, friends of a user have to access the user's profile from farther replica points which increases the average access cost. However, even though *MPD* chose same number of replicas as *MNR* (i.e., 1), it chose different replicas which inadvertently lead to a decreased access cost. *MAC* achieves the lowest access cost but with increased message overhead (as shown in Figure 6.15b). Figure 6.15b shows that message overhead increases with increase in replication degree.

6.9 Related Work

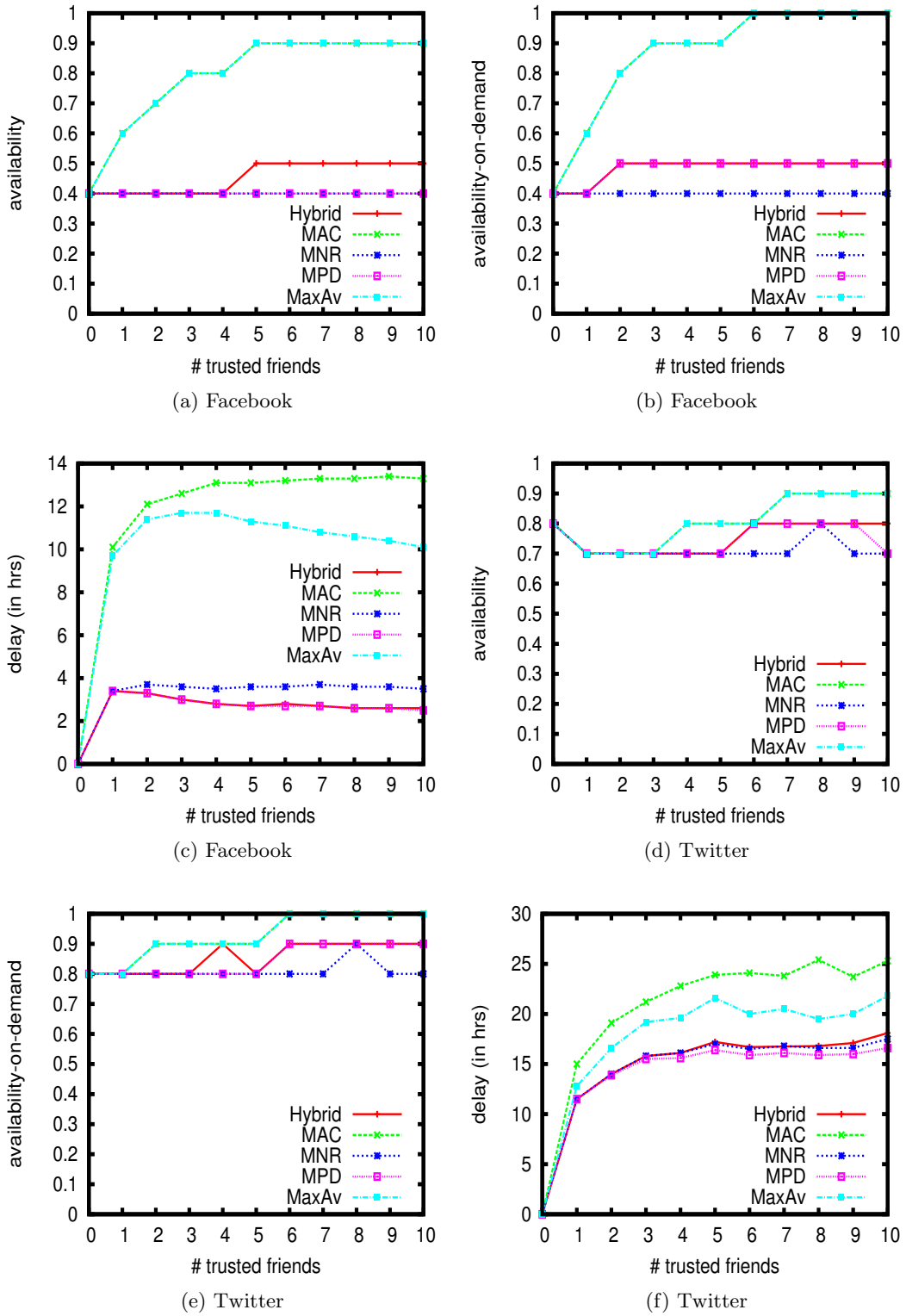
There is a significant related work on privacy issues in social networks. The issue of using decentralized infrastructures for organizing OSNs in a privacy-preserving manner, was addressed by the research community [45], [50], [114] which is explained in detail in Chapter 2. A decentralized OSN, Vis-à-Vis is proposed in [114], where, a user's profile content is stored at his own machine called as virtual individual server (VIS). VISs self-organize into P2P overlays, one overlay per social group that has access to content stored on a VIS. Three different storage environments are considered: cloud alone, P2P storage on top of desktops, a hybrid storage, and their availability, cost, and privacy trade-offs were studied. In the desktop-only storage model, a *socially-informed replication scheme* was proposed, where a user replicates his content to his friend nodes and delegates access control to them. However, normally, a uses trusts only a fraction of his friends to the extent of delegating access control enforcement, as considered in our *My3* approach along with online time information.

The authors in [108] pursue the notion of online times for a P2P client in detail. Various replica placement strategies are studied analytically. The Diaspora [19] project aims to build a user-owned decentralized online social network. It consists of independently owned *pods* (or servers) which host user profiles and form the network. However, the Diaspora system needs the pods to be online always. We believe that the *My3* model for decentralized OSNs where users can run their clients for a fraction of time compared to always-on availability of Diaspora, is more amenable.

To the best of our knowledge, *My3* is the first system that uniquely identifies the availability of decentralized OSNs as the critical concern for their adaptability and considers user behavior characteristics on OSNs and exploit them in its design.

6.10 Conclusion

In this chapter, we presented the design of *My3*, a privacy-preserving decentralized OSN. We evaluated its performance regarding different evaluation criteria using real world data traces. As experimentally found, high availability is achievable (close to 1) with a tolerable profile replication factor (4-5) for appropriate replication algorithm ($MaxAv$). The *My3* system performs very well for the availability-on-demand metric, which is a more critical performance metric for OSNs than the availability. The availability-on-demand reaches 1 for a low replication factor despite a very low online time of the clients. Thus the *My3* system could be a viable decentralized alternative to centralized infrastructures. We also demonstrated that the system can meet personalized performance objectives.



(a) Facebook (b) Facebook

(c) Facebook (d) Twitter

(e) Twitter (f) Twitter

Figure 6.9: Average of all users.

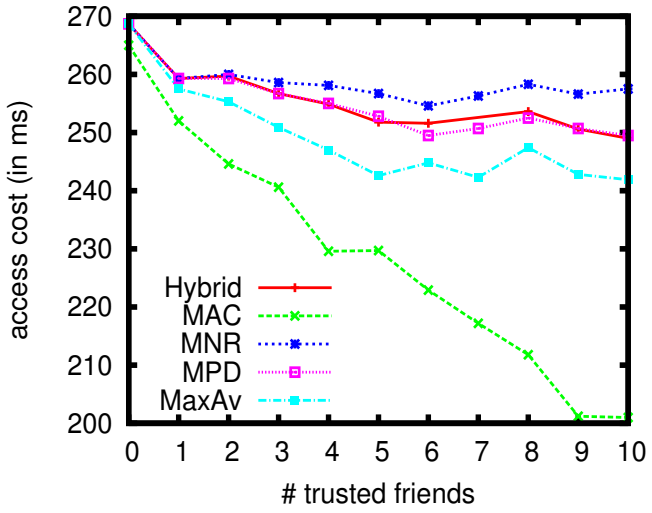
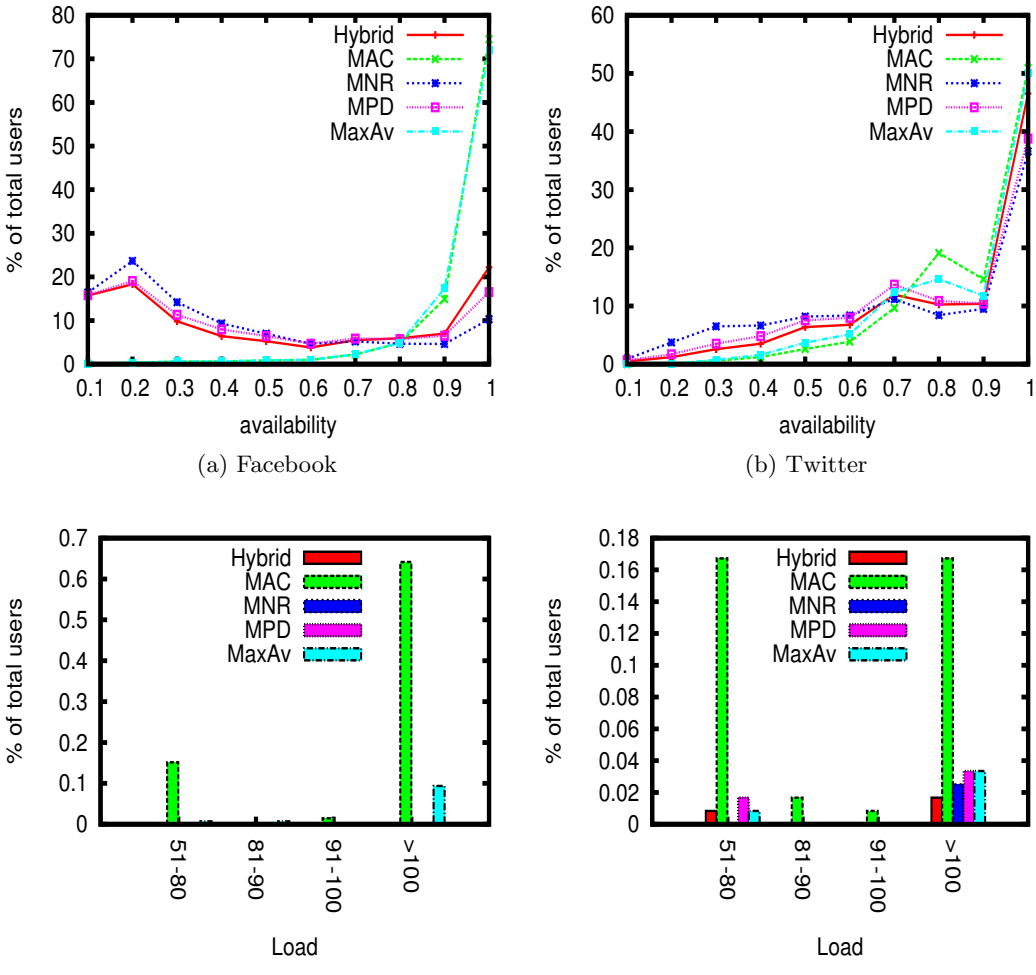


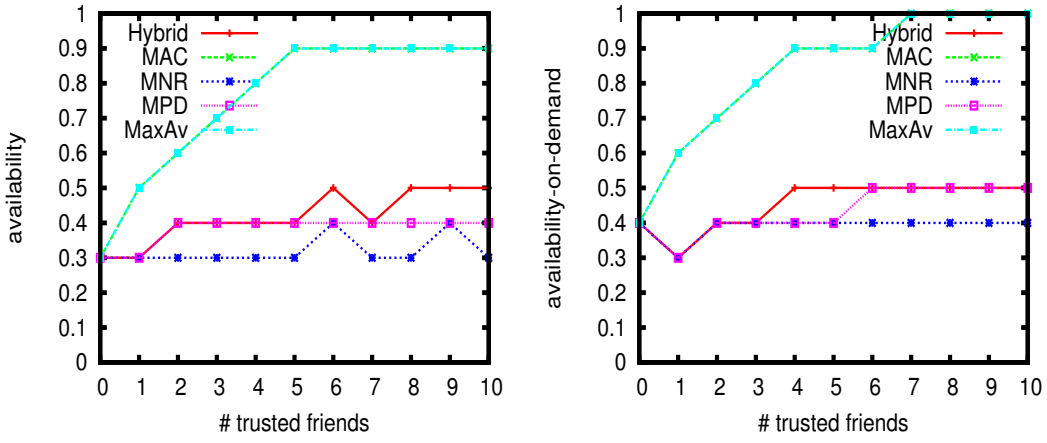
Figure 6.10: Twitter average of all users.



(a) Facebook (b) Twitter

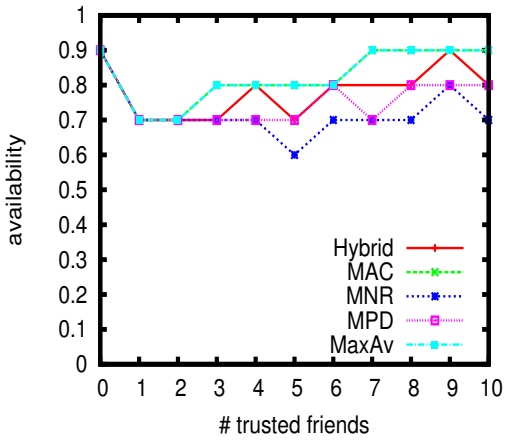
(c) Facebook (d) Twitter

Figure 6.11: Distribution of availability and load.

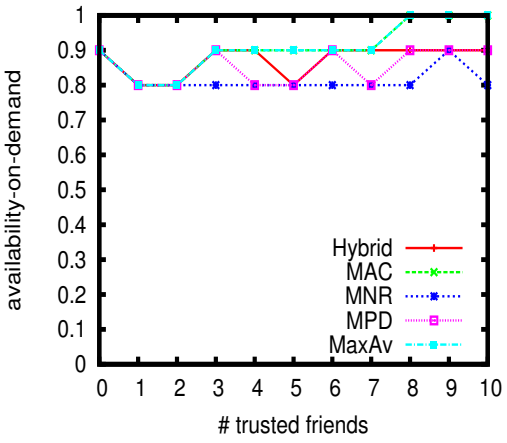


(a) Facebook

(b) Facebook

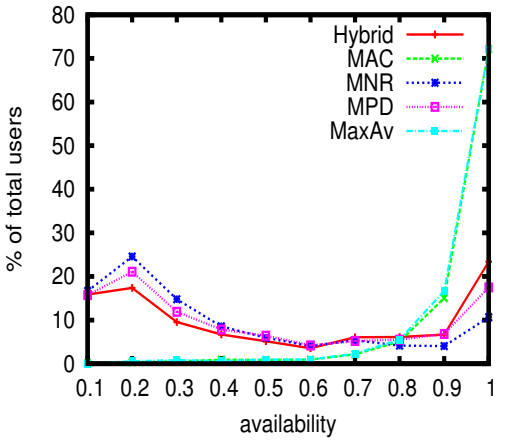


(c) Twitter

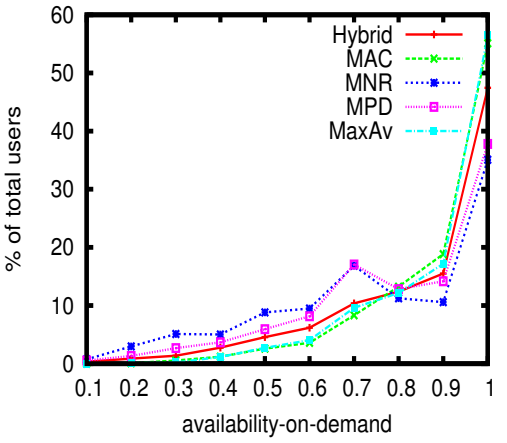


(d) Twitter

Figure 6.12: Average of users with degree 20.

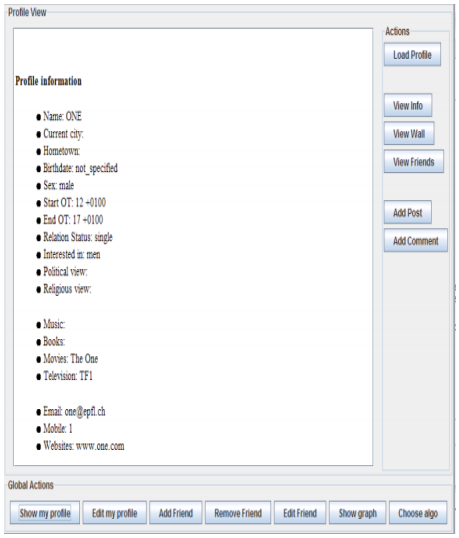


(a) Facebook

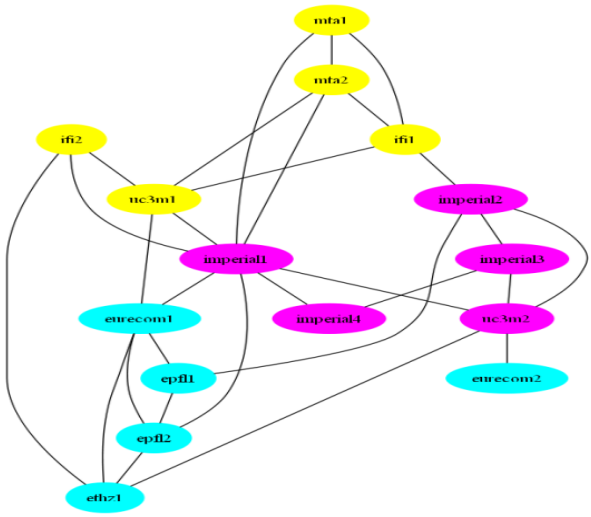


(b) Twitter

Figure 6.13: Distribution of availability: User- level replica choices.

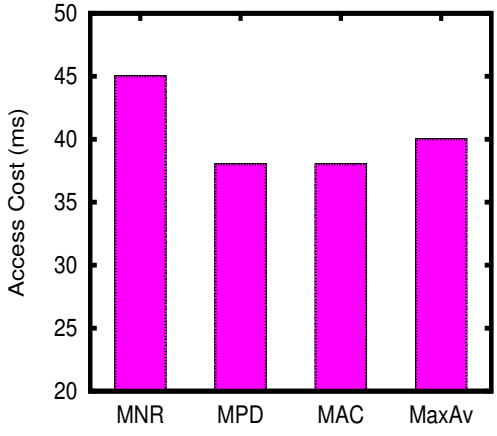


(a) Screenshot of *My3* prototype user interface.

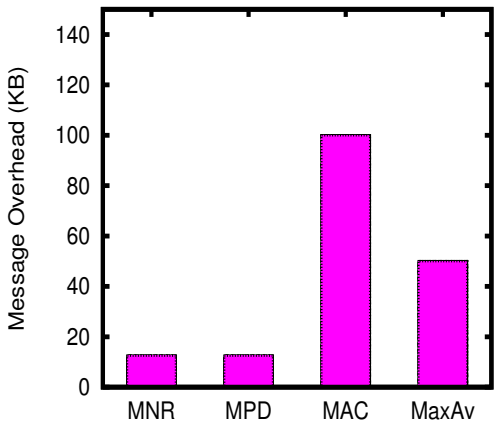


(b) The social graph used to configure the *My3* prototype. The colors differentiate the users respective time zones.

Figure 6.14: *My3* prototype experiments graph.



(a) Access Cost in ms



(b) Message Overhead in KBs

Figure 6.15: Results from experiments on *My3* prototype.

Part V

Conclusion

Chapter 7

Conclusion

The little that has been done
should not blind us to do the
vast that remains to be done.

S. Radhakrishnan

7.1 Summary of the Work

The amount of private and semi-private data has exploded on the Internet into unprecedented scales as the Web 2.0 paradigm becomes ubiquitous. Controlling the accesses on such sensitive data and protecting the data privacy, recently gained critical importance. In this thesis, we addressed this issue in detail. First, we dealt with the problem of controlling access to data resources published in a multi-publisher peer-to-peer environment as dictated by the owners of the data. We emphasized the fact that there exist multiple locations in the network where the user-defined access control policy can be enforced. We proposed several overlay architectures that help to enforce fine-grained access control [95]. The proposed mechanisms provide the publishers, not only the ability to control the granularity of the access-controlled content, but also the flexibility to adopt heterogeneous access-control models to coexist in the system simultaneously. Based on the granularity of an access-controlled item, an access control aware P2P system can be categorized as a Level-0 to a Level-2 system. In *controlled-query (CQ)* approach, the access control is exercised during query phase. In *controlled-reply (CR) approach*, it is done during query reply time. We proposed two CR-approach based techniques namely *trusted sub-overlays (CR-TSO)* and *trusted proxy set (CR-TPS)* and discussed the merits of each, in detail. A P2P semantic data sharing application is demonstrated where one of the proposed mechanisms, namely the CQ-based approach is used for access control enforcement.

We recognized the problem of building search-efficient access controlled P2P content publishing systems and proposed *PANACEA* [100, 101], a novel indexing mechanism which, as demonstrated in the thesis, significantly improves the search performance. We

highlighted the privacy characteristics the resulting index should possess given that such an index need to be hosted on untrusted peers. *PANACEA* preserves the resource and provider privacy of the content shared. It offers tunable knobs to the publishers, who can control the degree of privacy offered by the resulting index. In addition, a detailed probabilistic and information-theoretic analysis of the privacy offered by the system, was also provided. Efficiency requirements for such privacy preserving indexing mechanisms were charted out and *PANACEA* was compared in detail with the existing state-of-the-art methods including the most privacy-preserving and the most search-efficient P2P alternatives.

In this thesis, we dealt with the problem of privacy in today's online social networks and proposed *My3*, a privacy-preserving decentralized OSN [96, 97]. In its design, *My3* exploits unique characteristics of OSN workloads such as the locality of user OSN profile accesses and unique content access timings and patterns. The real world trust among friends on the social network is considered while choosing replication points for the social network content. We proposed different performance evaluation metrics for decentralized OSNs, namely, availability, availability-on-demand, update propagation delay, replication degree, access cost, and replication load [98]. *My3* replication algorithms optimize one or more of these performance metrics. These algorithms take a user's set of trusted friends and their online times as input, and output a set of replication points. We evaluated *My3*'s performance using real world data traces from Facebook and Twitter. As experimentally found, high availability is achievable with a tolerable profile replication factor [99]. We also demonstrated that the *My3* system can meet user-defined personalized performance objectives.

7.2 Future Work

We recognize that the work described in this thesis can be strengthened in a number of ways and we suggest the following as future work.

7.2.1 *ACPeer*

In relation to the *ACPeer* systems (Chapter 3), sophisticated cryptography mechanisms need to be investigated for efficient implementation of the *CQ-based* approach w.r.t the number of encryption and decryption keys needed to enforce an access control policy configuration. The prototype described in 3.4 involves a huge number of cryptographic keys, which is in the order of the number of subjects present in the system. Achieving a lower number of encryption keys simplifies both: the key management and key revocation when access control policy is changed. Several performance evaluation studies need to be performed on the prototype to measure the effectiveness of the proposed approaches.

7.2.2 *PANACEA*

In relation to the *PANACEA* system (Chapter 4):

- The proposed approach should accommodate content replication. In many settings, the content may not be stored on a single peer/publisher. It will be replicated to improve the availability. Indexing such access-controlled content replicated on multiple peers must be addressed. One of the solutions we envisage includes a group of trusted peers (much similar to a TPS described in Section 3.3.3.2) replicating content on each other and participating in the index as a single provider. In this case, a reply for any query on the index contains one or more of such group of peers. The searcher has to consult the peers in each group separately to retrieve the resource queried for.

- Secondly, the analytical formulation done in Section 4.4 can be improved. For some of the parameters in the formulation, closed formulas were not presented (for example, the term $P^{U}_{V,a}$). Note that, in the evaluation (Section 4.5), the values for these parameters are derived from the simulation experiments.

- In addition to resource and provider privacies, other privacy concerns arising in a multi-publisher context should be explored and *PANACEA* design should be adopted to meet these privacy concerns as well. For example, privacy preserving indexing of the content of a resource, which would allow querying for contents of a file. Requirements of such an enhanced application can not be met by mere resource and provider privacies offered by current design of *PANACEA*.

- As a future work, we intend to improve the *PANACEA* prototype built on top of the Kademlia protocol.

7.2.3 *My3*

In relation to the *My3* system (Chapter 6), we propose to consider the following as future study:

- The current *My3* system needs to be rigorously studied against encryption based decentralized online social network designs. The encryption based solutions can replicate users' profile content across the P2P system and thus increase the availability but introduce a huge overhead in the form of complicated cryptographic key management.

- *Fault tolerance* should be integrated into the *My3* design. In the current design, if one replica goes offline, the user profile will not be accessible if at least one of the other replicas is not online. *My3* replication algorithms always try to minimize the number of replicas chosen. Hence, no multiple replicas that will be online simultaneously, are chosen for replication. However, in a distributed system, nodes

are taken granted to fail or become unavailable from time to time. One of the fault tolerant solutions is to encrypt the user's profile and store in the P2P overlay (in addition to the replication on trusted peers as done in *My3*.) More trusted friends should be chosen as replicas in order to have more than one replica to be online at any given point of time.

- The online time models proposed in Section 6.7.3.1 need to be studied further. We plan to explore the system behavior for richer online time models especially considering the user degrees in the OSNs given that, a high degree node tends to stay online longer. Predicting user online presence behaviors and exploiting them to cleverly select replication points is a challenging problem.

- *My3* also involves dealing with access control policies, identity management and data integrity; which we leave for future work.

- Update propagation delay, defined in Section 5.2.3.3, should be as low as possible, for any decentralized OSN. Realizing a third party infrastructure and allowing update messages among the replicas to be exchanged via this infrastructure keeps updates readily available to replicas when they come online. It minimizes the delay significantly, however, needs a secured and privacy preserving way to store and forward messages by an untrusted third party. Such an infrastructure can be used to store the user's profile content as well, which can serve accesses when all the replicas are offline.

- The *PANACEA* indexing mechanism can be used to share social profile content hosted in *My3* in a privacy preserving way. Most of existing conventional online social networks severely lack the ability to search the content hosted inside users profiles (for example, Facebook). To retrieve a data item shared by one of a user's friends, the user has to perform linear search over all the items shared by the friend. This seriously limits the OSN's usability. The index of *PANACEA* enables efficient search and simultaneously meets the critical privacy concerns of the OSN users.

- We plan to undertake enhancing the *My3* prototype beyond Planetlab testbed and deploy it on workstations and mobile phones.

Bibliography

[1] *http://en.wikipedia.org/wiki/Dominating_set.* xxii, 128

[2] *http://www.bittorrent.com/.* 3

[3] *http://cassandra.apache.org/.* 3

[4] *http://en.wikipedia.org/wiki/Yahoo!_Mash.* 4

[5] *http://en.wikipedia.org/wiki/Bolt_(website).* 4

[6] *http://en.wikipedia.org/wiki/Bahu.* 4

[7] *http://bits.blogs.nytimes.com/2012/09/04/hackers-claim-to-have-12-million-apple-device-records/.* 23

[8] *http://bits.blogs.nytimes.com/2012/06/06/linkedin-was-breached-now-what/.* 23

[9] *http://bits.blogs.nytimes.com/2012/07/12/yahoo-breach-extends-beyond-yahoo-to-gmail-hotmail-aol-users/.* 23

[10] *http://www.businessinsider.com/warning-google-buzz-has-a-huge-privacy-flaw-2010-2.* 24

[11] *http://en.wikipedia.org/wiki/Facebook_Beacon.* 24

[12] *http://bits.blogs.nytimes.com/2011/04/02/the-rsa-hack-how-they-did-it/.* 24

[13] *http://www.mozilla.org/en-US/collusion/.* 24

[14] *http://bits.blogs.nytimes.com/2012/08/14/hulu-faces-privacy-test-in-court/.* 25

[15] *http://www.nytimes.com/2012/08/16/technology/germans-reopen-facebook-privacy-inquiry.html.* 25

[16] *http://www.technolog.msnbc.msn.com/technology/technolog/fbi-surveillance-backdoor-might-be-open-hackers-947887.* 25

[17] *http://www.engadget.com/2012/08/16/facebook-starts-really-truly-deleting-removed-photos/.* 25

[18] *http://gigaom.com/2012/08/15/friends-can-share-your-facebook-profile-with-the-government-court-rules/.* 25

[19] *http://diasporaproject.org/.* 28, 101, 122, 140

[20] *http://code.google.com/p/jkad/.* 96

[21] *http://www.emule-project.net/home/perl/general.cgi?l=1.* 96

[22] *http://www.planet-lab.org/.* 96, 139

[23] Anger for path social network after privacy breach. *http://bits.blogs.nytimes.com/2012/02/12/disruptions-so-many-apologies-so-much-data-mining/.* 24

[24] The data deluge. *http://www.economist.com/node/15579717.* 3

[25] Ftc charges deceptive privacy practices in google's rollout of its buzz social network. *http://www.ftc.gov/opa/2011/03/google.shtm.* 25

[26] Gnutella. *http://www.gnutella.com.* 12, 13

[27] Instagram 3.0 new maps feature: A privacy wake-up call? *http://www.wired.com/gadgetlab/2012/08/instagram-3-0-update-maps/.* 25

[28] Ip latency statistics. *http://www.verizonbusiness.com/about/network/latency/.* 124, 135

[29] Napster. *http://www.napster.com.* 12

[30] Nepomuk: The social semantic desktop. *http://nepomuk.semanticdesktop.org/.* 13, 33

[31] The p-grid project. *http://www.p-grid.org/index.html.* 34

[32] Price of facebook privacy? start clicking. *http://www.nytimes.com/2010/05/13/technology/personaltech/13basics.html.* 4, 24

[33] Ssl specification. *http://wp.netscape.com/eng/ssl3/.* 34

[34] TEAM: Tightening knowledge sharing in distributed software communities by applying semantic technologies. *http://www.team-project.eu/.* 6, 13, 33, 39, 53

[35] Tor: Anonymity online. *www.torproject.org.* 20, 98

[36] Wuala- the p2p storage. *http://wua.la/en/home.html.* 12, 33

[37] K. Aberer. P-Grid: A self-organizing access structure for P2P information systems. *Sixth International Conference on Cooperative Information Systems (CoopIS 2001), Lecture Notes in Computer Science*, 2172:179–194, 2001. 12, 33, 55

[38] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt. GridVine: Building internet-scale semantic overlay networks. In *International Semantic Web Conference (ISWC)*, volume 3298 of *LNCS*, pages 107–121, 2004. 13, 33, 34

[39] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. Farsite: federated, available, and reliable storage for an incompletely trusted environment. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, pages 1–14, New York, NY, USA, 2002. ACM. 12, 18

[40] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-molina, K. Kenthapadi, N. Mishra, R. Motwani, U. Srivastava, D. Thomas, J. Widom, and Y. Xu. Vision paper: Enabling privacy for the paranoids. In *In: Proceedings of VLDB 2004*, pages 708–719, 2004. 20, 98

[41] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. In *Proc. of the ACM SIGCOMM*, 2009. 26

[42] M. Bawa, R. J. Bayardo, Jr, R. Agrawal, and J. Vaidya. Privacy-preserving indexing of documents on the network. *The VLDB Journal*, 18(4):837–856, 2009. 4, 6, 21, 22, 69, 97, 98

[43] M. Bawa, J. Roberto J. Bayardo, and R. Agrawal. Privacy-preserving indexing of documents on the network. In *vldb'2003: Proceedings of the 29th international conference on Very large data bases*, pages 922–933. VLDB Endowment, 2003. 38, 48, 89, 90

[44] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user behavior in online social networks. In *Proc. of the IMC*, 2009. 110, 123, 133

[45] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta. Peerson: P2p social networking: early experiences and insights. In *Proc. of the ACM EuroSys Workshop on Social Network Systems*, 2009. 3, 27, 101, 140

[46] M. Cai and M. Frank. Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 650–657, New York, NY, USA, 2004. ACM. 13, 33

[47] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: a distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, pages 15–15, Berkeley, CA, USA, 2006. USENIX Association. 3

[48] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46–67, 2001. 12, 20, 21, 41, 63, 97, 98

[49] B. Crispo, S. Sivasubramanian, P. Mazzoleni, and E. Bertino. P-hera: Scalable fine-grained access control for p2p infrastructures. In *ICPADS '05: Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, pages 585–591, Washington, DC, USA, 2005. IEEE Computer Society. 3, 19, 21, 59, 97

[50] L. A. Cutillo, R. Molva, and T. Strufe. Privacy preserving social networking through decentralization. In *Proc. of the WONS*, 2009. 27, 125, 140

[51] J. F. da Silva, L. P. Gaspary, M. P. Barcellos, and A. Detsch. Policy-based access control in peer-to-peer grid systems. In *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 107–113, Washington, DC, USA, 2005. IEEE Computer Society. 18, 59

[52] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 202–215, New York, NY, USA, 2001. ACM. 3, 12, 33

[53] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15(429-444):2–1, 1977. 64

[54] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *21st ACM Symposium on Operating Systems Principles, SOSP'07*, October 2007. 3, 12

[55] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *Proc. of PET*, San Francisco, CA, USA, April 2002. 78

[56] R. Dingledine, M. J. Freedman, and D. Molnar. The free haven project: distributed anonymous storage service. In *International workshop on Designing privacy enhancing technologies*, pages 67–95, New York, NY, USA, 2001. Springer-Verlag New York, Inc. 21, 98

[57] A. Felt and D. Evans. Privacy protection for social networking apis. In *Proc. of the W2SP*, 2008. 24

[58] P. Fenkam, S. Dustdar, E. Kirda, G. Reif, and H. Gall. Towards an access control system for mobile peer-to-peer collaborative environments. In *WETICE '02: Proceedings of the 11th IEEE International Workshops on Enabling Technologies*, pages 95–102, Washington, DC, USA, 2002. IEEE Computer Society. 18, 59

[59] A. Fiat and M. Naor. Broadcast encryption. In D. Stinson, editor, *Advances in Cryptology ? CRYPTO? 93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer Berlin / Heidelberg, 1994. 26

[60] P. Fraigniaud and P. Gauron. D2b: a de bruijn based content-addressable network. *Theor. Comput. Sci.*, 355(1):65–79, Apr. 2006. 12

[61] J. L. Frank. F2f: reliable storage in open networks. In *Proc. of IPTPS'06*. 118

[62] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4), 2010. 20, 24, 98

[63] C. Fung. Collaborative intrusion detection networks and insider attacks. 2(1):63 – 74, 2011. 125

[64] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, and W. Kellerer. Outtweeting the Twitterers - Predicting Information Cascades in Microblogs. In *Proc. of the WOSN*, 2010. 108, 109, 131

[65] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.*, 63(3):241–263, Mar. 2006. 14

[66] E. Goh, H. Shacham, N. Modadugu, and D. Boneh. SiRiUS: Securing Remote Untrusted Storage. In *10th Annual Network and Distributed System Security Symposium (NDSS)*, 2003. 18

[67] K. Graffi, P. Mukherjee, B. Menges, D. Hartung, A. Kovacevic, and R. Steinmetz. Practical security in p2p-based social networks. In *Proc. of the IEEE LCN*, October 2009. 28

[68] Grunbacher and Nuremberg. Posix access control lists on linux. *http://www.suse.de/ agruen/acl/linux-acls/online*. 34

[69] S. Guha, N. Daswani, and R. Jain. An experimental study of the skype peer-to-peer voip system, 2006. 110, 133

[70] S. Guha, K. Tang, and P. Francis. Noyb: privacy in online social networks. In *Proc. of the WOSP*, Seattle, WA, USA, 2008. 23, 26, 98, 101, 121

[71] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: a scalable overlay network with practical locality properties. In *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, pages 9–9, Berkeley, CA, USA, 2003. USENIX Association. 45

[72] P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009. 13, 55

[73] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving p2p data sharing with oneswarm. *Technical report, University of Washington*, 2009. 20

[74] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 29–42, Berkeley, CA, USA, 2003. USENIX Association. 18

[75] M. Karnstedt, K.-U. Sattler, M. Richtarsky, J. Mller, M. Hauswirth, R. Schmidt, and R. John. Unistore: Querying a dht-based universal storage. In *ICDE*, pages 1503–1504. IEEE, 2007. 13, 33

[76] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards ip geolocation using delay and topology measurements. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 71–84, New York, NY, USA, 2006. ACM. 135

[77] D. Kifer and J. Gehrke. l-diversity: Privacy beyond k-anonymity. In *In ICDE*, page 24, 2006. 20

[78] Y. Kim, D. Mazzocchi, and G. Tsudik. Admission control in peer groups. In *NCA '03: Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, page 131, Washington, DC, USA, 2003. IEEE Computer Society. 41

[79] B. Krishnamurthy and C. E. Wills. On the leakage of personally identifiable information via online social networks. In *Proc. of the WOSN*, 2009. 23, 24, 101, 121

[80] I.-F. Lam, K.-T. Chen, and L.-J. Chen. Involuntary information leakage in social network services. In *Proc. of the 3rd International Workshop on Security*, 2008. xxv, 24, 25, 101

[81] J. Li, M. Krohn, D. Mazières, and D. Shasha. Secure untrusted data repository (sundr). In *OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*, pages 9–9, Berkeley, CA, USA, 2004. USENIX Association. 18

[82] N. Li and T. Li. t-closeness: Privacy beyond k-anonymity and ?-diversity. In *In Proc. of IEEE 23rd Int?l Conf. on Data Engineering (ICDE?07*, 2007. 20

[83] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard. A cooperative internet backup scheme. In *ATEC '03: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 3–3, Berkeley, CA, USA, 2003. USENIX Association. 12, 33

[84] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove. Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, Proc. of the IMC, pages 61–70, New York, NY, USA, 2011. ACM. 4, 24

[85] M. M. Lucas and N. Borisov. Flybynight: mitigating the privacy risks of social networking. In *Proc. of the WPES*, 2008. 26, 101

[86] T. Luu, F. Klemm, I. Podnar, M. Rajman, and K. Aberer. Alvis peers: a scalable full-text peer-to-peer retrieval engine. In *P2PIR '06: Proceedings of the international workshop on Information retrieval in peer-to-peer networks*, pages 41–48, New York, NY, USA, 2006. ACM. 38

[87] M. Madejski, M. Johnson, and S. M. Bellovin. The failure of online social network privacy settings. Technical Report CUCS-010-11, Department of Computer Science, Columbia University, February 2011. 24

[88] M. Marcon, B. Viswanath, M. Cha, and K. P. Gummadi. Sharing social content from home: a measurement-driven feasibility study. In *Proceedings of the 21st international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '11, New York, NY, USA, 2011. ACM. 27

[89] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Proc. of IPTPS*, 2002. 3, 12, 15, 63, 64, 96

[90] B. McBride. Jena: A semantic web toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002. 55

[91] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proc. of the 7th Internet measurements conference*, 2007. 23, 121

[92] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 251–260, New York, NY, USA, 2010. ACM. 24

[93] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips. Tribler: a social-based peer-to-peer system: Research articles. *Concurr. Comput. : Pract. Exper.*, 20(2):127–138, 2008. 28

[94] N. Rammohan. A note on secure p-grid. *Technical Report, LSIR-REPORT-2008-005, Swiss Federal Institute of Technology (EPFL)*. 34

[95] N. Rammohan, Z. Miklos, and K. Aberer. Towards access control aware p2p data management systems. In *2nd International workshop on data management in peer-to-peer systems (EDBT workshops)*, 2009. 4, 21, 63, 149

[96] N. Rammohan., T. Papaioannou, and K. Aberer. Privacy-aware and highly-available osn profiles. In *Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on*, pages 211 –216, june 2010. 150

[97] N. Rammohan., T. Papaioannou, and K. Aberer. My3: A highly-available p2p-based online social network. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 166 –167, 31 2011-sept. 2 2011. 150

[98] N. Rammohan., T. Papaioannou, and K. Aberer. Towards the realization of decentralized online social networks: An empirical study. In *32nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 155 –162, june 2012. 131, 135, 150

[99] N. Rammohan, T. G. Papaioannou, and K. Aberer. A decentralized online social network with efficient user-driven replication. In *IEEE International Conference on Social Computing (SocialCom)*, 2012. 150

[100] N. Rammohan, T. G. Papaioannou, and K. Aberer. Privacy preserving indexing of access controlled data in peer-to-peer systems. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, to be submitted. 149

[101] N. Rammohan, T. G. Papaioannou, Z. Miklos, and K. Aberer. Tunable privacy for access controlled data in peer-to-peer systems. In *22nd International Teletraffic Congress (ITC 22)*, 2010. 149

[102] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *ACM SIGCOMM 2001*, August 2001. 12, 33

[103] M. K. Reiter and A. D. Rubin. Crowds: anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998. 20, 21, 64, 68, 73, 98

[104] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. Opendht: a public dht service and its uses. *SIGCOMM Comput. Commun. Rev.*, 35(4):73–84, 2005. 124

[105] C. Riederer, V. Erramilli, A. Chaintreau, B. Krishnamurthy, and P. Rodriguez. For sale : your data: by : you. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 13:1–13:6, New York, NY, USA, 2011. ACM. 26

[106] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–340, 2001. 12, 33

[107] L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, and K.-I. Ko. A greedy approximation for minimum connected dominating sets. *Theoretical Computer Science*, 329(1-3):325 – 330, 2004. 127

[108] K. Rzadca, A. Datta, and S. Buchegger. Replica placement in p2p storage: Complexity and game theoretic analyses. In *Proc. of the ICDCS*, June 2010. 140

[109] R. S. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40–48, 1994. 3, 16, 17

[110] N. Saxena, G. Tsudik, and J. H. Yi. Admission control in peer-to-peer: design and performance evaluation. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 104–113, New York, NY, USA, 2003. ACM. 41

[111] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger. Understanding online social network usage from a network perspective. In *Proc. of the IMC*, New York, NY, USA, 2009. 110, 123, 133

[112] B. Schneier. Applied cryptography, second edition: Protocols, algorithms, and source code in c. 1996. 42

[113] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proc. of PET*, San Francisco, CA, USA, April 2002. 78

[114] A. Shakimov, A. Varshavsky, L. P. Cox, and R. Cáceres. Privacy, cost, and availability tradeoffs in decentralized osns. In *Proc. of the WOSN*, 2009. xxi, 27, 28, 118, 140

[115] Sharma, R. et. al. An empirical study of availability in friend-to-friend storage systems. In *In Proc. of P2P'11*. 118

[116] A. Singh. Secure management of networked storage services: Models and techniques. *Ph.D Thesis, College of Computing, Georgia Institute of Technology, 2007.* 17, 18, 38, 39

[117] A. Singh, B. Gedik, and L. Ling. Agyaat: Mutual anonymity over structured p2p networks. *Emerald Internet Research Journal*, 16(2), 2006. 20, 21, 98

[118] A. Singh, M. Srivatsa, and L. Liu. Efficient and secure search of enterprise file systems. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 18–25. IEEE Computer Society, July 2007. 36

[119] J. Sonia, N. Shirin, M. Prateek, B. Nikita, and K. Apu. Decent: A decentralized architecture for enforcing privacy in online social networks. In *Proc. of the SESOC*, 2012. 27

[120] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001. 4, 12, 14, 33

[121] J. Sun, X. Zhu, and Y. Fang. A privacy-preserving scheme for online social networks with efficient revocation. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010. 26

[122] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5), 2002. 67

[123] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: better privacy for social networks. In *Proc. of the CoNEXT*, 2009. 26

[124] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *Proc. of the WOSN*, 2009. 108, 131

[125] K. Watanabe, Y. Nakajima, N. Hayashibara, T. Enokido, M. Takizawa, and S. M. Deen. Trustworthiness of peers based on access control in peer-to-peer overlay networks. In *ICDCSW '06: Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems*, page 74, Washington, DC, USA, 2006. IEEE Computer Society. 18, 59

[126] Wilson et. al. User interactions in social networks and their implications. In *Proc of the EuroSys '09*. 111

[127] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 543–554. VLDB Endowment, 2007. 79

[128] Yahoo! Research. Pnuts- platform for nimble universal table storage. *http://research.yahoo.com/node/212*. 3, 12

[129] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 267–278, New York, NY, USA, 2006. ACM. 89

[130] S. Zerr and W. Nejdl. Privacy preserving document indexing infrastructure for a distributed environment. *Proc. VLDB Endow.*, 2008. 20, 98

[131] Y. Zhang, X. Li, J. Huai, and Y. Liu. Access control in peer-to-peer collaborative systems. In *ICDCSW '05: Proceedings of the First International Workshop on Mobility in Peer-to-Peer Systems (MPPS) (ICDCSW'05)*, pages 835–840, Washington, DC, USA, 2005. IEEE Computer Society. 18, 59

[132] E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 531–540, New York, NY, USA, 2009. ACM. 24

Rammohan Narendula, Ph.D.

Strengths

– Experience in building distributed systems especially privacy friendly distributed social networks and multi-party data publishing systems in untrusted environments.
– Experience in managing the complete life cycle of a software product.

Education

2007–2012 **Ph.D.**, ***EPFL, Switzerland***.
– School of Computer and Communication Sciences.
– Distributed Information Systems Lab (LSIR).
– *Thesis :* Privacy-aware Publishing of Decentralized Access-Controlled Content.
– *CGPA :* 5.5/6
– *2007–2009 :* Component Architect, TEAM project, European Union.

2002–2004 **M.S. (Research)**, ***Indian Institute of Technology Madras, India***.
– Dept. of Computer Science and Engineering.
– High Performance Computing and Networking Lab (HPCN).
– *Thesis :* On Cost-Efficient QoS Routing of Multicast Traffic in Wavelength Routed WDM networks.
– *CGPA :* 9.8/10
– *Academic Distinction :* Topper in the batch.

1998–2002 **B.Tech.**, ***Kakatiya University, Andhra Pradesh, India***.
– Dept. of Computer Science and Engineering.
– *Academic Distinction :* University first rank.

Work Experience

Summer-2009 **Research Intern**, ***HP Research Labs***, **Bangalore, India**.
– Securing work-flows spanning multiple organizations.
– *Team size :* 1

2007-2009 **Component Architect**, ***TEAM project, European Union***, **EPFL**.
– *TEAM- Tightening knowledge sharing in distributed software communities by applying semantic technologies.*
  – Enables a network of users to mutually share the knowledge and experience for overall improvement of the traditional software development processes.
  – "Sensors" installed on a user's computer, automatically capture this knowledge.
  – Knowledge represented using semantic technologies.
– `http://www.teamweaver.org/wiki/index.php/Main_Page`
– *Role :* Architecture design and development of distributed metadata storage with access-controlled sharing.
– *Team size :* 1
– *Technologies :* Java, OSGI, Jena, RDF, OWL, X.509, RSA.

2005-2006 **Software Engineer**, ***Qualcomm***, **Hyderabad, India**.
– Design and development of Mobile TV (*DVB-H* protocol) engine.
– *Team size :* 4
– *Technologies :* C++, DVB-H, Lint.

Project Experience

My3- *A decentralized online social network :*
– A functional prototype is implemented in Java and deployed on PlanetLab.
– A Java-based My3 simulator justifies the merits of the distributed storage infrastructure using Facebook and Twitter datasets.
– *Technologies :* Java, JUNG (Java Universal Network Graph) framework, PlanetLab, Perl.

PANACEA- *Privacy preserving indexing of access controlled content in an untrusted multi-publisher environment :*
- A working prototype based on Kademlia client is built and deployed on PlanetLab.
- A Java-based simulator demonstrates its enhanced performance over the state-of-the-art.
- *Technologies :* Java, Mathematica, JKad, PlanetLab, Perl.

TEAM- *Tightening knowledge sharing in distributed software communities by applying semantic technologies :*
- Architecture design and development of distributed metadata storage with access-controlled sharing.

AWS- *Business case study of Amazon Web Services (AWS) :*
- The business and economic principles that drove the evolution of AWS are analyzed.
- Several promising future growth trajectories were proposed.

TickIt- *An Android app for event ticket management :* It offers an all-in-one ticketing solution to generate and distribute secured e-tickets and control event entry check-in with multiple devices using a single Android app.

Professional Experience

- **Principal Investigator**, *EPFL Site, PlanetLab*, `www.planet-lab.eu`.
- **Teaching Assistant**, *Distributed Information Systems course* and *Programmation C.*
- **Project Supervisor**, *6 M.S. and 2 Bachelor student projects.*
- **Reviewer**, *IEEE Transactions on Parallel and Distributed Systems, Computer Networks Journal, ICDCS-09, MDM-10, P2P-08, P2P-12, EDBT-12, VLDB-13.*

Areas of Interest

Privacy and Security, Trust, Social networking, Network routing, Data analytics, Distributed systems, Data management systems.

Research Work

My Ph.D thesis broadly addresses :
- The privacy and efficiency issues in an untrusted multi-publisher environment.
  - A novel indexing mechanism that enables efficient searches and simultaneously protects the privacy of the content, from unauthorized users.
  - Tools to quantify the privacy achieved.
- Privacy issues in today's online social networks.
  - A distributed storage infrastructure for decentralized social networks, that helps users to share social network updates without compromising their privacy.
  - Exploits unique characteristics of social network workloads in its design, which are abstracted from Facebook and Twitter datasets.

Selected Publications

- N. Rammohan, T. G. Papaioannou, and K. Aberer. A Decentralized Online Social Network with Efficient User-Driven Replication. *IEEE International Conference on Social Computing (SocialCom), 2012.* [Acceptance rate : <10%]
- N. Rammohan, T. G. Papaioannou, and K. Aberer. Towards the Realization of Decentralized Online Social Networks : an Empirical Study. *4th IEEE ICDCS/HOTPOST, Macau, China, 2012.*
- B. Agir, T. G. Papaioannou, N. Rammohan, K. Aberer, and J.-P. Hubaux. Adaptive Personalized Privacy in Participatory Sensing. *5th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Tucson, Arizona, USA, 2012.*

– N. Rammohan, T. G. Papaioannou, and K. Aberer. My3 : A highly-available P2P-based online social network. *11th IEEE International Conference on Peer-to-Peer Computing, Japan, 2011.*
– N. Rammohan, T. G. Papaioannou, Z. Miklos, and K. Aberer. Tunable Privacy for Access Controlled Data in Peer-to-Peer Systems. *22nd International Teletraffic Congress (ITC 22), Amsterdam, 2010.*
– N. Rammohan and C. Siva Ram Murthy, On-Line Multicast Routing with QoS Constraints in WDM Networks with No Wavelength Converters. *Elsevier Computer Networks Journal, vol. 50, num. 18, p. 3666 - 3685, 2006.*
– Sushant Sinha, N. Rammohan, and C. Siva Ram Murthy, Dynamic Virtual Topology Reconfiguration Algorithms for Groomed WDM Networks, *Journal of Photonic Network Communication, vol. 9, no. 2, pp. 181-195, March 2005.*
– N. Rammohan, T. G. Papaioannou, and K. Aberer. On Privacy Preserving Indexing of Access Controlled Data in Untrusted Multi-Publisher Environments. *IEEE Transactions on Parallel and Distributed Systems (in submission).*
– B. Agir, T. G. Papaioannou, N. Rammohan, K. Aberer, and J.-P. Hubaux. User-side Adaptive Protection of Location Privacy in Participatory Sensing. *Submitted to 11th International Conference on Mobile Systems, Applications and Services (MobiSys 2013).*

*Detailed publication list is available on my homepage.

Misc.

– Won University Gold Medal for scoring highest marks in B.Tech considering all the disciplines offered. [2002]
– Placed 14th in All India Entrance Examination conducted by Indian Insti. of Technology, Roorkee. [2002]
– State level 16th rank in 12th Standard and 13th rank in 10th Standard (out of $\sim$ 1 million students). [1995-1998]
– Active executive member of Yuva, Indian Association, EPFL. [2000-2012]

Information

– **Mobile :** +41-78 62 144 62
– **Email :** rammohan.narendula@epfl.ch
– **Permanent Email :** n.rammohan@gmail.com
– **Homepage :** `http://people.epfl.ch/rammohan.narendula`
– **Google Scholar :** `http://bit.ly/PiUzIt`
– **LinkedIn :** `http://ch.linkedin.com/in/nrammohan`
– **Address :** Chemin du Veilloud 16, 1024 Ecublens, Switzerland.

– **Visas :** *Switzerland :* Working permit B, *USA :* B1.

References

– Available upon request.