An empirical study of availability in friend-to-friend storage systems

Rajesh Sharma and Anwitaman Datta
Nanyang Technological University, Singapore
raje0014@e.ntu.edu.sg, anwitaman@ntu.edu.sg

Matteo Dell'Amico and Pietro Michiardi
Eurecom, Sophia-Antipolis, France
{matteo.dell-amico,pietro.michiardi}@eurecom.fr

***Abstract*—*Friend-to-friend* networks, *i.e.* peer-to-peer networks where data are exchanged and stored solely through nodes owned by trusted users, can guarantee dependability, privacy and uncensorability by exploiting social trust. However, the limitation of storing data only on friends can come to the detriment of data availability: if no friends are online, then data stored in the system will not be accessible. In this work, we explore the trade-offs between redundancy (*i.e.*, how many copies of data are stored on friends), data placement (the choice of which friend nodes to store data on) and data availability (the probability of finding data online). We show that the problem of obtaining maximal availability while minimizing redundancy is NP-complete; in addition, we perform an exploratory study on data placement strategies, and we investigate their performance in terms of redundancy needed and availability obtained. By performing a trace-based evaluation, we show that nodes with as few as 10 friends can already obtain good availability levels.**

Keywords: friend-to-friend (F2F), storage systems, data placement, NP-complete, heuristics

I. Introduction

Peer-to-peer (P2P) storage systems have been studied for over a decade, starting with the OceanStore [5] project. The premise of P2P storage is crowdsourcing the storage cloud [2] to other end-users. One of the many design issues in such systems is the choice of peers at which to store data. A specific subclass of P2P storage systems have emerged based on the placement choice being constrained to 'friends' of the data owner, for example, FriendStore [7]. The basic characteristics of such friend-to-friend (F2F) storage systems are: (i) real-life social trust is exploited to guarantee a dependable system (*e.g.*, a friend of mine won't erase my data); (ii) data access is predominantly confined within a small social neighborhood. These networks, also known as 'darknets' when the focus is on security, can also guarantee privacy and resistance to censorship [1]. F2F storage thus constitutes a good building block for diverse applications such as personal backup service and decentralized online social networking.

For personal backup, while data persistence is more critical, data availability is nevertheless desirable. For decentralized online social networking systems such as SuperNova [6], availability is of paramount importance. Thus, a fundamental problem that arises is determining what kind of availability one can achieve in a storage system where data placement for any specific data owner is constrained by the use of only peer nodes run by friends of the data owner.

This work was supported in part by A*Star SERC grant 072 134 0055 and NTU/MoE Tier-1 grant RG 29/09. The collaboration between NTU and Eurecom was supported by Merlion grant.

There are several variations of this basic question that would interest a F2F storage system designer. A baseline is determined when all friends of a node store its data. This is the best in terms of availability that one can achieve subject to the constraint of using friend nodes exclusively. However, there are some obvious variations worth studying. Can the same availability (or any other predetermined threshold of availability) be achieved using only a subset of the node's friends? How does the law of diminishing returns work in terms of availability, as the number of used friends is increased? If a stipulated number of friends are to be used, what is the best availability that can be achieved? Furthermore, the way to measure availability itself may vary. For a personal backup application, the data owner may care for the data to be available only when it itself is online - for example, with other portable devices. For a decentralized online social networking application, the data owner can serve its own data when it is itself online, but will like the friends to make the data available when it is itself offline. More generally, availability may also be determined based on whether it was available when there was any access request for the data. These various interpretations of availability may depend on the access and application specific characteristics.

The achievable and achieved performance would depend on the (temporal) characteristics of individual nodes' egocentric networks (*i.e.*, the social network consisting of those nodes and their respective immediate friends), the actual data-placement policies determining a subset of friends to store data at, as well as the interpretation of availability itself. This paper is a first attempt to formalize these quantitative aspects of F2F storage systems, exploring algorithmic aspects of data placement in (sub-)optimal subset of friends, and exposition of the efficacy of F2F storage systems using trace-driven simulations using real egocentric social network traces capturing additionally node availability traces over time.

The important contributions of this paper include (*i*) defining some key characteristics of an ego-network which influence the achievable availability in a F2F storage system, (*ii*) observing that identification of a minimal set of friends to achieve the maximum achievable coverage is in fact analogous to the set cover problem, and hence NP-hard, (*iii*) propose greedy heuristic data placement algorithms, and (*iv*) evaluation of the heuristics using trace-driven simulation using real ego-network traces and accompanying up/down-time information.

II. Ego-Centric Network Characterization

In a F2F storage system using exclusively friends, the quality of service an individual node can have depends solely on its own ego-centric network.[1] We next define some key characteristics to define a node's ego-centric network based on the uptime/downtime characteristics of nodes.

For a node n, we use F_n to represent its set of friends. We study the system for a period of τ discrete time units (seconds, minutes, or other quanta), *i.e.*, the whole period of study can be represented as $T_{tot} = \{t_1, t_2, t_3 ..., t_\tau\}$. The uptime of a node n is represented with a time trace T_n where $t_i \in T_n$ iff node n is online at time t_i.

In general, a node may want to achieve availability of its data for an arbitrary subset of time $T'_n \subseteq T_{tot}$. For example, if a node n wants 24/7 availability then $T'_n = T_{tot}$, while if it wants availability of its data only for the periods when it is itself offline, then $T'_n = T_{tot} - T_n$.

Let T'_{F_n} represents the collection of time trace of all the friends n_j of node n, *i.e.*, $n_j \in F_n$. For notational conciseness, we use $\oint$ to represent $|F_n|$.

$$T'_{F_n} = \{T'_{F_{n_1}}, T'_{F_{n_2}}, T'_{F_{n_3}}, \dots, T'_{F_{n_\oint}}\} \tag{1}$$

A. Achievable Coverage

We define *achievable coverage* AC_n as the fraction of the total time for which a node n is able to get data availability by storing data at all and only its friends, out of the total time for which it is seeking data availability. $AC_n = |T^{ac}_n|/|T'_n|$ where

$$T^{ac}_n = \cup_{j=1}^{\oint} T'_{F_{n_j}} - \overline{T'_n} \tag{2}$$

B. Degree of Criticality

Out of the periods where coverage is achievable, there are time slots which can be covered using an unique neighbor, *i.e.*, there is only one neighbor which is up for that period, and thus this specific neighbor is critical in providing such coverage. We denote using T^{cr}_n the time slots which are critical.[2] We define the *degree of criticality* as $DC_n = |T^{cr}_n|/|T'_n|$. A high degree of criticality would typically imply that more friends are needed to achieve a particular level of availability, and also, the system is more vulnerable to faults. Conversely, low degree of criticality implies more redundancy, which translates into more flexible choices, and robustness of the resulting system.

III. Data Placement at Friends

Replicating data at all friends trivially allows a node to get maximal achievable coverage. However, this would result in very high costs for storing data in multiple copies, in particular for users with a large number of friends. An obvious question is to find the minimal number of friends that can provide the same maximal coverage. It is straightforward to show that this is in fact an NP-hard problem. This finding motivates us to propose heuristics that strive to optimize the trade-off between the number of copies and the achieved data availability.

[1] Load distribution at nodes may percolate over multiple social hops. Such load-related issues are ignored in this preliminary work.

[2] Note that $T^{cr}_n \subseteq T^{ac}_n$.

A. Maximal Coverage With Minimum Replication

Let us now focus on the problem of reaching the maximal coverage for a node n (*i.e.*, T^{ac}_n in Equation 2) while minimizing the number of friends employed. This corresponds to the *Set Cover problem* introduced and proven to be NP-hard by Karp in 1972 [4]: using the notation we used so far, Set Cover can be formalized as follows. Given the family of traces T'_{F_n} as defined in Equation 1, find the minimum value k such that a sub-family of traces $C \subseteq T'_{F_n}$ of cardinality k having $\bigcup C = T^{ac}_n$ exists. This observation of NP-hardness motivates us to design a heuristic greedy policy for data placement.

B. Greedy Data Placement Heuristics

Let F^{cr}_n be the set of critical neighbors of node n, *i.e.*, each of them covers at least one critical time slot. To get maximum possible coverage, in our greedy heuristic the node first picks all the friends n_j s.t $n_j \in F^{cr}_n$. If critical nodes are not able to cover all the T'_n time slots for which the node n is looking for coverage, then the node n picks a non-critical node that covers the maximum number of the time slots not yet covered. This process continues until there are no nodes left which cover further uncovered time slots, or there are no more nodes left.

Variations of this basic approach could entail (i) a cap on the quality of service, *e.g.*, a node may decide to settle for coverage not less than 85% of the maximum achievable coverage; or (ii) a cap on the maximum number of friends to be used, *e.g.*, a node may want to limit the replication factor of its data to five. We note that for such variants, other heuristics – possibly those which do not differentiate among critical and non-critical nodes – may yield better performance.

IV. Results

For our experiments, we carry out trace-driven simulations using instant messaging traces. Numerous social network graphs are available; likewise, several traces from P2P and other distributed systems exist, providing up/down-time of nodes. However, we could not find openly available real traces containing both social relation information and the availability trace of the same nodes. We thus obtained such dataset from the operator of an instant messaging (IM) server in Italy.

Our dataset suffers from various shortcomings. First, it does not have adequate geographic diversity which can be exploited to achieve better availability. Furthermore, instant messaging usage pattern may arguably be different from the usage pattern of P2P storage systems or decentralized social networking applications. Our data is also a partial view of the complete IM network (*i.e.*, nodes have some contacts registered on different servers whose uptime information is not available).

Notwithstanding these issues, the dataset allows us to experiment on real data with a trace of both social and temporal node behavior, justifying its usage in this empirical study.

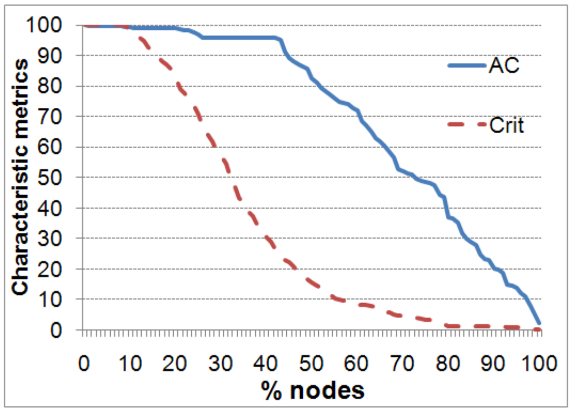

Fig. 1. Ego-centric network characteristics.

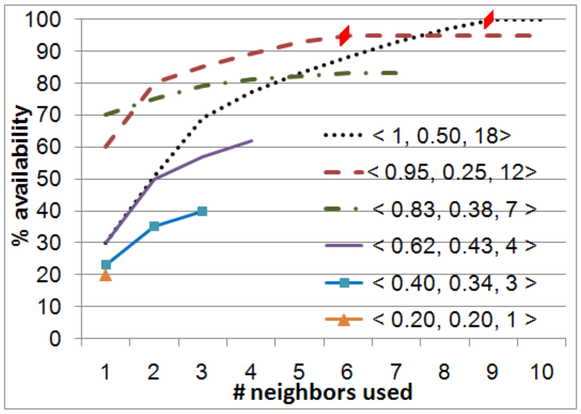

Fig. 2. Microscopic analysis.

In a friend-to-friend storage system, if all friends agree to cooperate, then the performance is likely to be better, since almost all the above mentioned drawbacks also happen to translate into more pessimistic estimates about the system environment. We will continue to look for more appropriate data sets in the future.

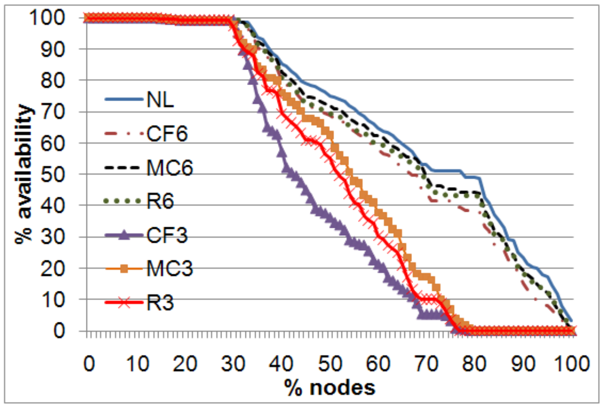

Fig. 3. Macroscopic analysis.

A. Experimental Results

The original trace comprised 3,436 nodes. After filtering the nodes that had no neighbors in the same server or were not part of the largest connected component of the social graph, we obtained a set of 848 unique nodes, each having between one and eighteen neighbors. Nodes' uptime/downtime behavior from one week was used to drive the data placement algorithm, and the achieved availability in the second week was studied. The rationale for such a simplistic setting is the well-known diurnal patterns [3] of individual users' uptimes.

In the following we summarize the characteristics of the ego-centric networks from our dataset and the system performance obtained using our data placement heuristics.

Ego-centric network characteristics: Figure 1 shows the characteristics of the nodes considered in our experiments, as derived from the whole period of two weeks of traces. A reverse CDF (cumulative distribution function) is plotted for the achievable coverage (*AC*). From this plot, we observe for instance, that roughly 50% nodes can achieve at least 90% of coverage (for the whole period, *i.e.*, if the ideal objective is to obtain 24/7 coverage for the two weeks studied) if they replicate their data at all their respective friends.

We likewise plot the reverse CDF graph showing the percentage of the total time where coverage is achieved using critical nodes (*Crit*), that is, only one friend was up during the corresponding amount of time. If this sole friend goes offline for whatever reason, then there are no alternatives for the corresponding time slots. Ideally, it is desirable for nodes to have a low value for this metric. We note that roughly 20% of nodes have more than 90% of their time covered by critical nodes - which highlights both the importance of critical nodes, as well as the potential vulnerabilities of a F2F storage system. However, we also observe that this plot has a relatively steeper slope (for instance, w.r.to *AC*). This is again a desirable trait, since it means many nodes can get coverage using noncritical nodes for a large amount of time.

Microscopic analysis: In Figure 2 we do a microscopic study of the performance of our data placement heuristic by looking at individual, representative ego-centric networks. We show five nodes in this plot, with distinct characteristics, summarized using the tuple $\langle AC, DC, \oint\rangle$, *i.e.*, achievable coverage, degree of criticality, and the number of neighbors. In these plots, we additionally get a glimpse of the marginal benefit of incrementally replicating at more friends.

Nodes with high number of neighbors, and relatively low degree of criticality, for instance, nodes represented with $\langle 1, 0.50, 18\rangle$, $\langle 0.95, 0.25, 12\rangle$ and $\langle 0.83, 0.38, 7\rangle$ enjoy good level of availability.[3] We note that, within these nodes, those with relatively lower criticality ($\langle 0.95, 0.25, 12\rangle$) obtain high level of availability with very few replicas, in contrast with a node with a relatively higher degree of criticality ($\langle 1, 0.50, 18\rangle$). There are however also greater overlaps among the neighbors' uptime, and thus, there is an effect of diminishing returns with additional replication, as the maxi-

[3]Recall that DC needs to be interpreted relative to the value of AC.

mum achievable availability is approached. Finally, when the maximum availability is indeed achieved, as indicated with a marker in some of these plots, there can obviously be no further benefit in adding more replicas. This corresponds to significant storage space savings with respect to a naive strategy which would replicate at all friends. For example, for the node with 18 neighbors, nine neighbors are enough to achieve 100% availability. In contrast, the rate of increase in the achieved availability with additional replication is higher for $\langle 0.62, 0.43, 4\rangle$ and $\langle 0.40, 0.34, 3\rangle$ which have fewer neighbors, and degree of criticality is relatively higher. We also show a node with only a single node, which naturally enjoys coverage for a very low period of time.

Macroscopic analysis: While one fundamental question is to minimize the number of replicas needed to achieve the maximum possible availability, an obvious dual to this question is, if a node decides to cap the degree of replication to a specific predetermined number, for instance due to bandwidth and storage budgets, what availability is achievable? The first heuristic algorithm, which necessarily uses the critical nodes first, may not be best suited to achieve maximum availability under a cap on the maximum number of neighbors that can be used. Under such a cap on the number of replicas, a greedy algorithm which does not discriminate between critical and non-critical neighbors is expected to perform best. As a base line for such scenarios with cap on the number of replicas, we study what is achieved when replication is done at randomly chosen neighbors instead.

We report our findings in Figure 3, which confirm these intuitions. CF3 and CF6 represent the results obtained using our heuristic which prefers *critical nodes first*, subject to replication cap of three and six respectively. MC3 and MC6 represent the results obtained using a greedy algorithm to *maximize coverage* subject to the corresponding caps, while R3 and R6 show the results obtained using *random* placement. NL represents the result obtained with our first heuristic with *no limit* on the degree of replication, hence minimizing the degree of replication subject to reaching the maximum achievable storage. As expected, MC outperforms the other strategies when availability needs to be maximized subject to a cap on the degree of replication. Interestingly, the randomized algorithm in fact outperforms the original heuristics under such limitation on replication, and achieves only slightly less availability than the greedy approach to maximize coverage. Finally, note that even using at most six friends yields only marginally less availability than the maximum achievable.

Further discussions: The data set we used was extremely sparse (most online social networks have much larger number of neighbors), which explains the low availability in many of the results. One may also argue that even if people have many more online friends, most of them are unlikely to share their storage spaces, and hence a usable network may also be sparser. In all cases, our experiments expose some interesting facets about F2F storage systems. Firstly, if a node has reasonably large number of friends (say, more than 10) willing to participate in a F2F storage system, then a very good level of availability is achievable, and our heuristic does a good job in determining a minimal subset of friends to achieve maximum coverage, saving unnecessary replication. However, if there is a limit on the allowed degree of replication, and the limit is small, then this heuristic suffers a drastic deterioration because of its preference to critical nodes. In contrast, a greedy strategy which does not discriminate between critical and non-critical nodes works well. More interestingly, even a randomized placement strategy works almost as well. We hypothesize that if there is a larger pool of nodes to choose from, it is likely that the greedy strategy will perform even better, while the random strategy's performance will deteriorate in comparison, since it will be more likely to pick some unsuitable nodes. Larger choice of nodes will also provide a bigger, and arguably temporally diverse pool to choose from - thus significantly increasing the availability.

V. Conclusion

In this work we performed an exploratory study on the trade-offs between redundancy, data placement and data availability in F2F storage applications. We have shown that the problem of finding a data placement with maximal availability minimizing redundancy is NP-complete, which prompted us to design heuristic data placement strategies. We then evaluated these policies through an empirical study based on a social network complemented with availability traces; in our case, some low availability results were due to the low number of connections in our particular trace; however, our results showed that reasonably good availability results could be obtained already by nodes that had around 10 friends. We also found out that – depending on the goal – different data placement heuristics are advisable. For example, if there is a limit on the applicable redundancy, a simple greedy heuristic performs better than placement strategies designed for minimizing replication for maximal coverage. We plan to continue working on this topic, by exploring the impact of limited storage resources on nodes, considering both global optimization techniques and interaction of selfish data placement strategies, and exploring more extensive traces.

References

[1] I. Clarke, O. Sandberg, M. Toseland, and V. Verendel. Private Communication Through a Network of Trusted Connections: The Dark Freenet. 2010. http://freenetproject.org/papers.html.

[2] A. Datta. Peer-to-peer storage systems: Crowdsourcing the storage cloud. In *ICDCN tutorial*, 2010.

[3] S. Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *IPTPS*, 2006.

[4] R.M. Karp. Reducibility among combinatorial problems. *Complexity of computer computations*, page 85, 1972.

[5] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *ACM SIGARCH Computer Architecture News*, volume 28, pages 190–201, 2000.

[6] R. Sharma and A. Datta. SuperNova: Super-peers Based Architecture for Decentralized Online Social Networks. Technical report, arXiv, 2011.

[7] D.N. Tran, F. Chiang, and J. Li. Friendstore: Cooperative online backup using trusted nodes. In *SocialNets '08: Proceedings of the 1st Workshop on Social Network Systems*, 2008.