The Case of Decentralized Online Social Networks

Rammohan Narendula
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—As the Online Social Networks (OSNs) amass unprecedented amounts of personal information, the privacy concerns gain considerable attention from the community. Apart from privacy-enabling approaches for existing OSNs, a number of initiatives towards building decentralized OSN infrastructures have emerged. However, before this paradigm becomes a serious alternative to current centralized infrastructures, some key design challenges, often conflicting with each other, have to be addressed. In this paper, we explore such design objectives concerning various system properties, namely availability, replication degree, user online times, privacy, and experimentally study the trade-offs among them based on real data sets from Facebook and Twitter. We introduce different mechanisms to model user online times in the OSN from their activity times. We demonstrate how different profile replica selection approaches significantly affect the system performance.**

***Keywords*-decentralized OSNs, privacy, empirical evaluation**

I. 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 great business value to the service provider, e.g., personalizing ads, but on the other hand, makes the users vulnerable to privacy breaches and malicious exploitation, e.g. burglars locating vacant houses. As a result, serious privacy concerns were raised in the past, by the research community [1], [2], [3]. Several proposals exist in the literature that aim to increase user privacy on the OSNs without altering the existing social network infrastructures, e.g. [4]. Alternatively, semi-/fully-decentralized OSN infrastructures such as, Peerson [5], My3 [6], [7], and Diaspora [8] are also pursued. 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 paper, we experimentally explore these trade-offs 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. To be explained later, an important parameter that affects all these metrics, is the *online time* of the user.

Since privacy is a serious concern in decentralized OSNs, in this paper, 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 in Facebook and Twitter, 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 efficiency.

The rest of the paper is organized as follows: Section II introduces various efficiency metrics for decentralized OSNs. In Section III, we deal with the replica placement strategies. Experimental methodology is discussed in Section IV followed by the results in Section V. Related work and conclusion are presented in Sections VI and VII, respectively.

II. The Context

A well-designed decentralized OSN application should promise user experience and functionality similar to that of existing centralized OSNs. A typical OSN allows its users to post messages or content onto his profile (like the "wall" in Facebook) or on other people's walls, send personal messages, chat with online friends, discover new friends, and retrieve 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.

A. Online Time Connectivity

Let OT_u denote the online time period of 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. Let NG_u be the set of his friends (i.e. neighbors) in the social graph. Assume that the profile of user u is replicated at some friends[1] $R_u \subseteq NG_u$. The profile of user u is accessible by an arbitrary user v 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 in time or unconnected. 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 $UnconRep$), replicas have to communicate among themselves using a third-party storage or a content delivery network

[1]In our study, all friends of a user are assumed to be trusted for hosting the user's profile replica.

(CDN). A decentralized OSN inherently privacy-conscious, should adopt the $ConRep$ approach for the replica selection.

B. 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:

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 guaranty on data consistency. We believe that a requirement of *eventual consistency* would be adequate for decentralized OSNs. Addressing the problem of consistency in detail is beyond the scope of the paper. 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.

2) Privacy requirements: Typically in a privacy-aware OSN, semi-private part of 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) poses 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 is vulnerable to security attacks by malicious users in the system. In addition, enforcing access control using encrypted content is inefficient.

C. Efficiency Metrics

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

1) Availability: The fraction of time in a day, a user's profile is accessible through the replicas. 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.

2) Availability-on-Demand: This metric quantifies the accessibility of the profile for the friends of a user. We introduce two variations of the metric: *Availability-on-Demand w.r.t 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. Second, *Availability-On-Demand w.r.t Activity*: Fraction of the times there was an activity on a user's profile in a specific time interval in the past and the profile was available.

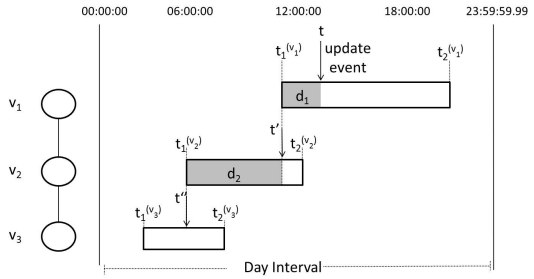


Fig. 1: Update propagation delay from v_1 to v_3.

3) Update Propagation Delay: The latency between the end 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 between them. In the case of $ConRep$, 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, with weight of the edge set to the update propagation delay between the two nodes. Updates among replicas are propagated via a multi-hop shortest path on this graph.

We explain the calculation of this delay in the example of Figure 1. In this figure, we assume three replicas of a certain profile residing at nodes v_1, v_2, and v_3 with different continuous online times represented with begin and end, $OT_{v_1} = [t_1^{(v_1)}, t_2^{(v_1)}]$, $OT_{v_2} = [t_1^{(v_2)}, t_2^{(v_2)}]$, $OT_{v_3} = [t_1^{(v_3)}, t_2^{(v_3)}]$, for which the online time graph is shown in the figure. Let an update event happen at replica v_1 at t, i.e. at the end of its online period. Then, this update would be communicated to v_2 at time t', which would take at least $24 - d_1$ hours, where d_1 is size of the overlap 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 time of at least $24 - d_2$ 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.

The *Update Propagation Delay* for a user u 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 graph. 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 actual delay for node v_2 is

4) Replication Degree: It is the number of replicas hosting a user's profile. This metric expresses the storage and com-

munication overhead involved in replicating the user's profile. Moreover, it can be seen as a degree of potential privacy breach of user's 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.

III. Replica Selection Policies

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 which are described in detail in the following:

A. 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 can be computed a priori as $\mid \cup_{f \in V(u)} OT_f \mid$. 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 $\cup_{f \in V(u)} OT_f$. 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 $ConRep$ 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.

The replica selection algorithm for maximizing the availability-on-demand w.r.t activity (resp. availability-on-demand w.r.t time) is again modeled as a set cover problem where the universe is the union of the activity times of all friends observed during a pre-defined time in the past (resp. union of online times of all the friends).

B. Most active friends as replicas (MostActive)

This approach prioritizes the most active friends 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 (time) 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 in a predefined-time frame in the past, are chosen as replicas. In case, there are no sufficient number of friends with non-zero activity, random friends are chosen.

C. Random friends (Random)

In this approach, friends of the user are randomly chosen to place the user profile replicas, which should be connected in time in the case of $ConRep$.

IV. 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 III, based on real data traces from Facebook and Twitter.

A. 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 [9] and a Twitter dataset [10].

The activity considered were the wall posts (for Facebook dataset), 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.

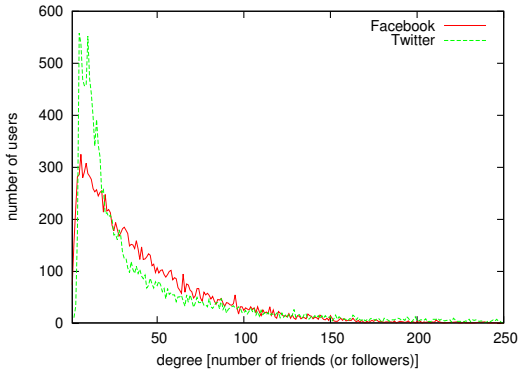


Fig. 2: Degree distributions of Facebook and Twitter datasets

1) Facebook: The Facebook dataset employed is the NewOrleans Network dataset [9], which has a total of 63,731 users creating a total of 876,994 wall-posts. A wall-post 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.

2) Twitter: We employed a simplified version of the Twitter dataset of [10], 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 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. 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).

B. 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 III. Then, all user activities are replayed in the system and the efficiency is measured according in terms of the metrics specified in Section II-C, as the replication degree is varied. The user online times are approximated by applying different models as explained in 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 $ConRep$ 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 w.r.t time*. 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, 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 w.r.t 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.

C. User online time models

As mentioned earlier, users online time is an important metric that affect 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:

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. As found for Orkut in [11], most active users stay online for more than an hour in a session, while 22% of the sessions last less than 20 minutes. To this end, we employed a fixed session length of 20 minutes, as a conservative choice.

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 [12].

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.

Out of all, we believe that Sporadic is the most realistic as it approximates online times very close to that of the real-world.

D. 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 IV-C. Nevertheless, we believe that, since the considered activities constitute the majority of the overall activities in Facebook and Twitter [13], our datasets can be considered representative for obtaining results of general applicability. Third, the datasets considered are small in size w.r.t the size of the OSNs Facebook and Twitter. However, the Facebook dataset considered covers more than 70% of the regional network [9].

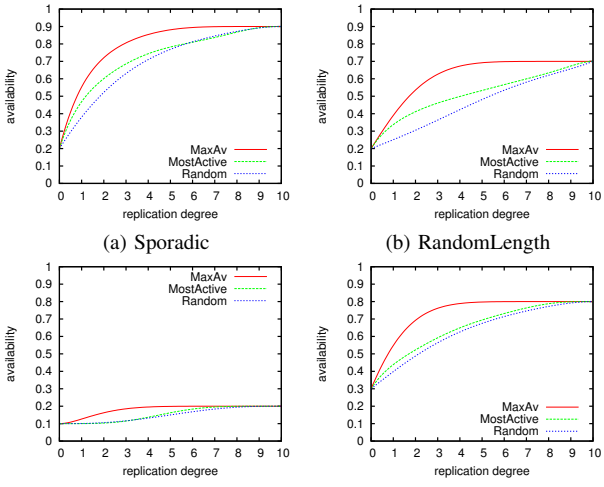
V. Results

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 IV-C. For the sake of clarity of presentation, we have smoothed the plots using Bezier curves to emphasize the different trends. Without loss of generality, 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:~ 300 and Twitter: ~ 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 and $RandomLength$ for online time, are repeated 5 times and averages are presented. Availability is computed as the fraction of number of distinct online hours (resp. minutes for Sporadic model) of replicas over 24 hours (resp. 1440 minutes), while the availability-on-demand w.r.t time is the fraction of number of distinct online hours of replicas over that of his friends.

A. Facebook

1) Availability vs. Replication degree: Availability increases with replication degree as is illustrated in Figure 3 and Figure 4 for the cases of connected and unconnected replicas respectively, for all online time models. As expected, $MaxAv$ 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 Figures 4c, 4d for the $FixedLength$ case.



(a) Sporadic (b) RandomLength

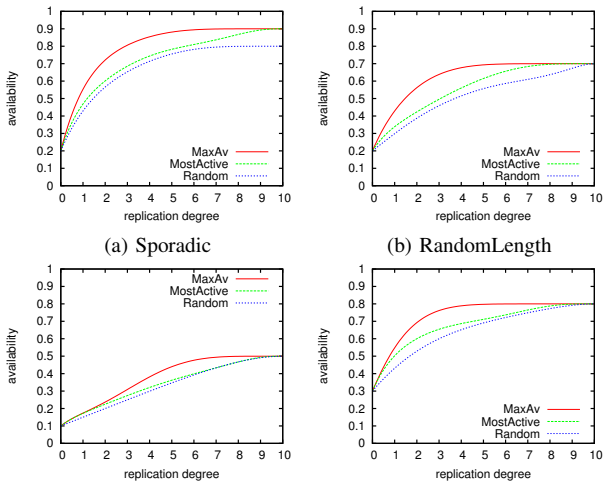
(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 3: Facebook-ConRep: Availability

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 the $Sporadic$), as shown in Figure 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 7 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. For the case of $UnconRep$, it is even higher.

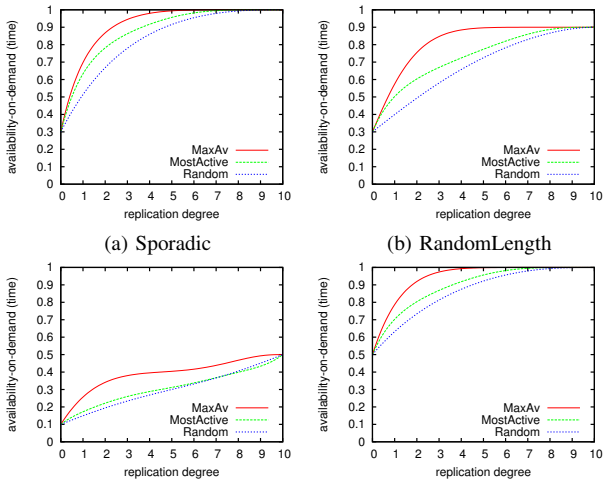
3) Update Propagation Delay vs. Replication Degree: Nonintuitively, the update propagation delay increases with



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 4: Facebook-UnconRep: Availability

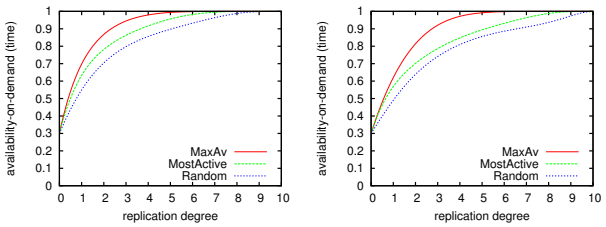


(a) Sporadic (b) RandomLength

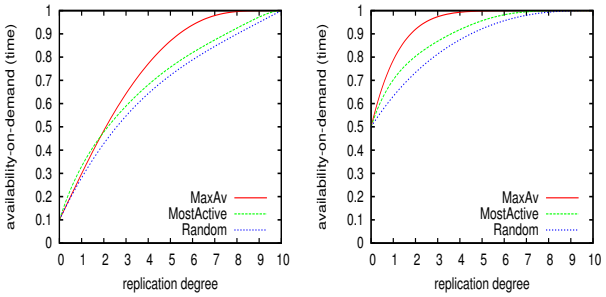
(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 5: Facebook-ConRep: Availability-on-Demand w.r.t Time

the replication degree, as depicted in Figure 9. However, this can be understood, as explained in Section II-C3, 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 the $Sporadic$ 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 II-C3) would be much lower. The delay is expected to be lower for $UnconRep$ case, as external communication means are used for update propagation.

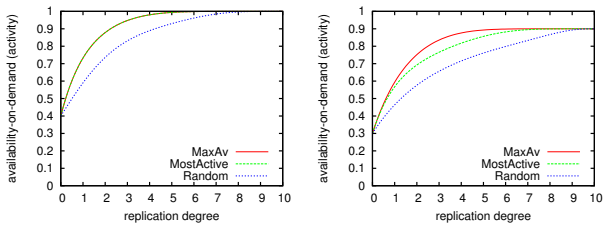


(a) Sporadic (b) RandomLength

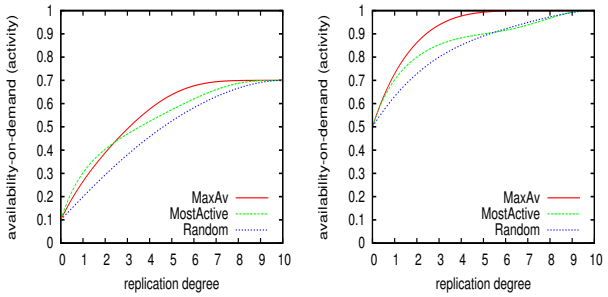


(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 6: Facebook-UnconRep: Availability-on-Demand w.r.t Time

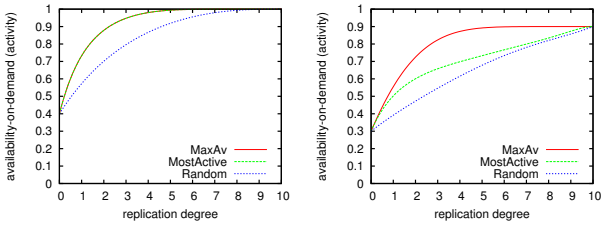


(a) Sporadic (b) RandomLength

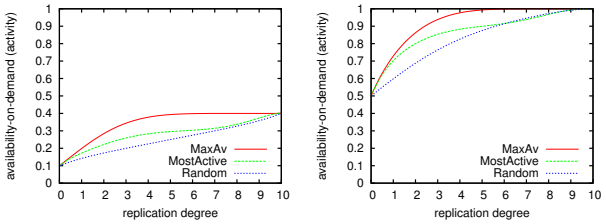


(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 8: Facebook-UnconRep: Availability-on-Demand w.r.t Activity

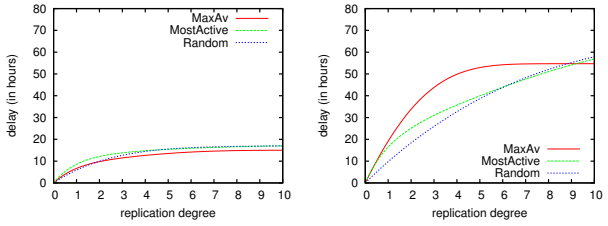


(a) Sporadic (b) RandomLength

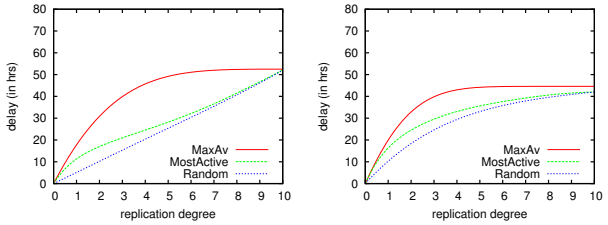


(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 7: Facebook-ConRep: Availability-on-Demand w.r.t Activity



(a) Sporadic (b) RandomLength



(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 9: Facebook-ConRep: Update Propagation Delay

B. Twitter

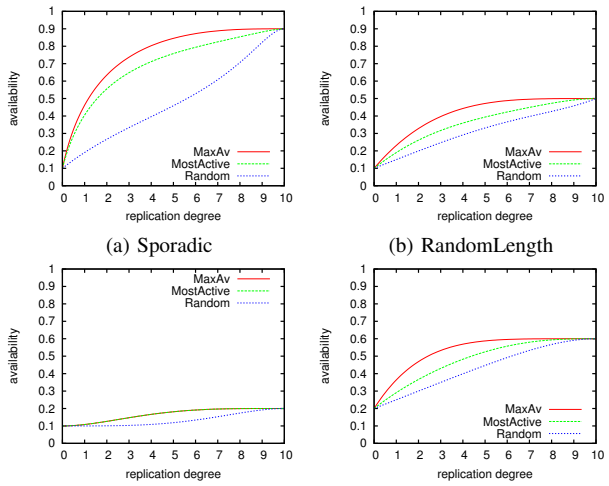
We observed similar results for the Twitter dataset for all the metrics. The plots for the availability metric are presented in Figure 10.

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

Also, note that, ideally higher availability-on-demand (time and 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.

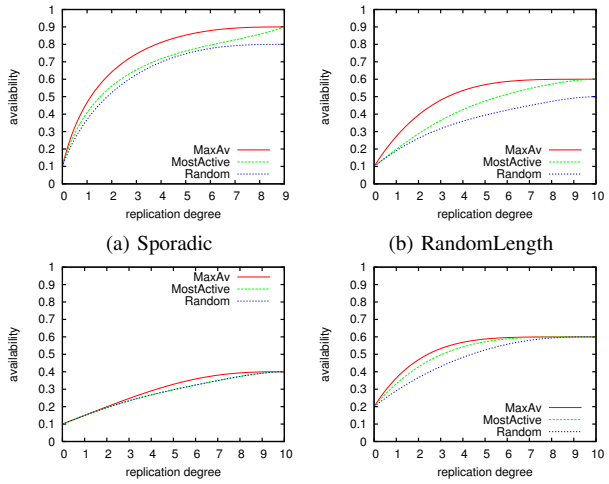
Also, $MostActive$ replica placement is a promising approach for decentralized OSNs as it is computationally simpler and does not require knowledge of the user online times, as opposed to $MaxAv$. Activities of friends and online time



(a) Sporadic (b) RandomLength

(c) UniformModel (2hours) (d) UniformModel (8hours)

Fig. 10: Twitter-ConRep: Availability



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

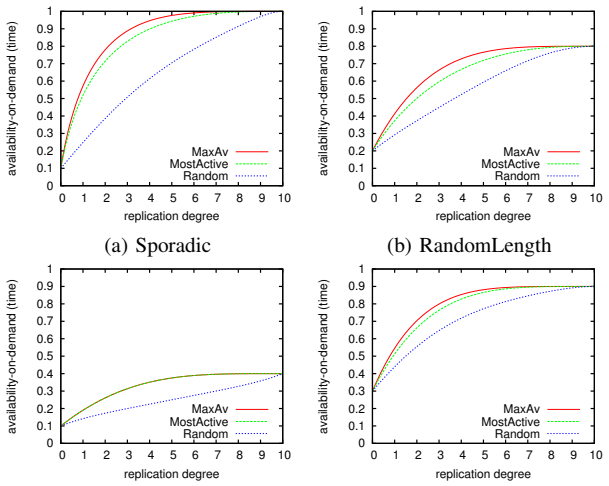
Fig. 11: Twitter-UnconRep: Availability

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.

The update propagation delay seems to be a big challenge towards the realization of decentralized OSNs; we empirically found delays of $\sim$2 days for some online time models. Although, the observed delay would be lower, it 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. CDN, DHT, cloud storage etc.) for exchanging updates. However, this would require encryption of the updates exchanged.

VI. Related Work

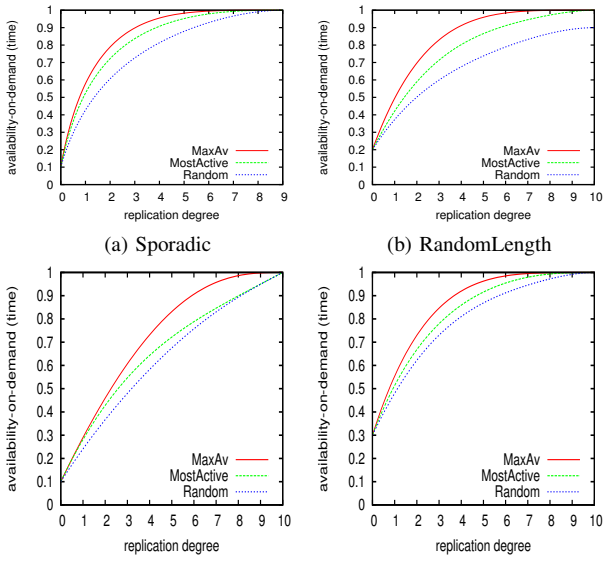
In the literature, there are many proposals on privacy-aware decentralized social networks [5], [6], [7], [8], [15].



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 12: Twitter-ConRep: Availability-on-Demand w.r.t Time

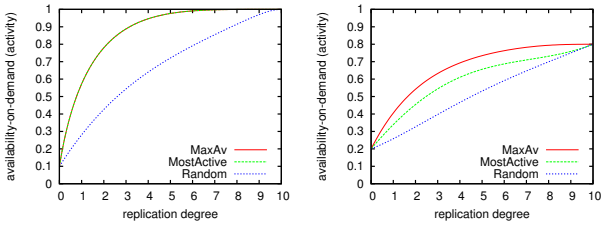


(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 13: Twitter-UnconRep: Availability-on-Demand w.r.t Time

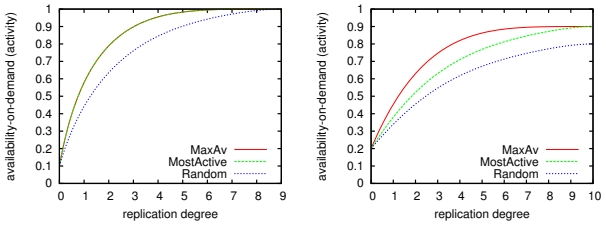
PeerSon [5] adopts encryption mechanisms for content storage and access control enforcement. [15] addresses privacy in OSNs by storing encrypted profile content in a P2P storage infrastructure. Each user in the OSN defines his own view ("matryoshka") of the system. In this view, nodes are organized in concentric rings, having nodes at each ring trusted by the nodes in 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. Also, LifeSocial [16] 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. However, they do not aim at



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

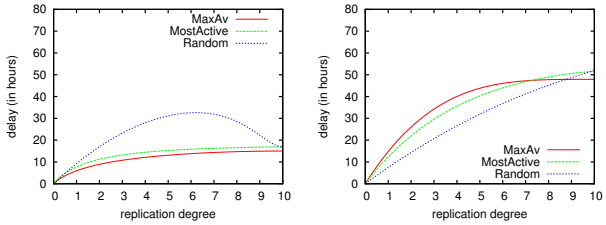
Fig. 14: Twitter-ConRep: Availability-on-Demand w.r.t Activity



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 15: Twitter-UnconRep: Availability-on-Demand w.r.t Activity



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 16: Twitter-ConRep: Update Propagation Delay

experimental evaluation of availability or other performance metrics. In [17], 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. Tribler [18] 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. In [6], [7], we have dealt with the design of a high-available decentralized OSN system. However, these works deal mainly with the replication placement assuming a single simple online time model, as opposed to the empirical present study of the main performance aspects of such a system. Finally, although still under development, Diaspora [8] is currently a decentralized OSN prototype system where each user maintains his profile available through a locally-hosted web server.

The authors in [19], [20] deal with friend-to-friend storage systems and our work complements to them. The work in [19] justifies that F2F systems are more reliable alternatives over conventional P2P systems storage by providing analytical and experimental evaluation. A more recent empirical study of availability of F2F systems is pursued in [20], which uses a dataset of an instant messaging service. Our study systematically analyzes the challenges for realizing the decentralized OSNs and employs data traces from two real and well-known OSN applications: Facebook and Twitter. We also consider two separate cases of connected and unconnected replicas. In a completely untrusted environment, OceanStore [21] proposed a persistent storage infrastructure through redundancy and cryptographic techniques. However, approaches that employ encryption mechanisms for access control and content storage involve complicated key management issues (e.g. key distribution, key revocation, key loss, etc.) and are highly inefficient in terms of storage overhead. Encryption mechanisms should be employed as well in the case of decentralized OSN when replication among friendly nodes cannot satisfy the desired availability level.

Lockr system [22] 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 [23] 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 NOYB approach [4] 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. On the other hand, flyByNight [24] encrypts the users' content that hosts on the OSN.

VII. Conclusion and Future work

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. As a future work, we want to pursue further empirical evaluations involving varying session lengths in the sporadic time model and larger datasets, by mining diurnal patterns of user behavior on the social network portals.

References

[1] 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.

[2] B. Krishnamurthy and C. E. Wills, "On the leakage of personally identifiable information via online social networks," in *Proc. of the WOSN*, 2009.

[3] A. Acquisti and R. Gross, "Imagined communities: Awareness, information sharing, and privacy on the facebook," in *Proc. of the PET*, 2006.

[4] S. Guha, K. Tang, and P. Francis, "Noyb: privacy in online social networks," in *Proc. of the WOSP*, Seattle, WA, USA, 2008.

[5] 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.

[6] 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*.

[7] ——, "My3: A highly-available p2p-based online social network," in *Proc. of P2P'11*.

[8] Diaspora, http://diasporafoundation.org/.

[9] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proc of the WOSN '09*.

[10] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, and W. Kellerer, "Outtweeting the Twitterers - Predicting Information Cascades in Microblogs," in *Proc of the WOSN '10*.

[11] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing user behavior in online social networks," in *Proc of the IMC '09*.

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

[13] Wilson et. al, "User interactions in social networks and their implications," in *Proc of the EuroSys '09*.

[14] R. Narendula, T. G. Papaioannou, and K. Aberer, "The case of decentralized osns," 2012, EPFL Technical Report.

[15] L. A. Cutillo, R. Molva, and T. Strufe, "Privacy preserving social networking through decentralization," in *Proc. of the WONS*, 2009.

[16] 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.

[17] Shakimov et. al, "Privacy, cost, and availability tradeoffs in decentralized osns," in *Proc. of the WOSN'09*.

[18] 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.*, vol. 20, no. 2, 2008.

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

[20] Sharma, R. et. al, "An empirical study of availability in friend-to-friend storage systems," in *In Proc. of P2P'11*.

[21] J. K. et. al, "Oceanstore: An architecture for global-scale persistent storage," in *In Proc. of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*.

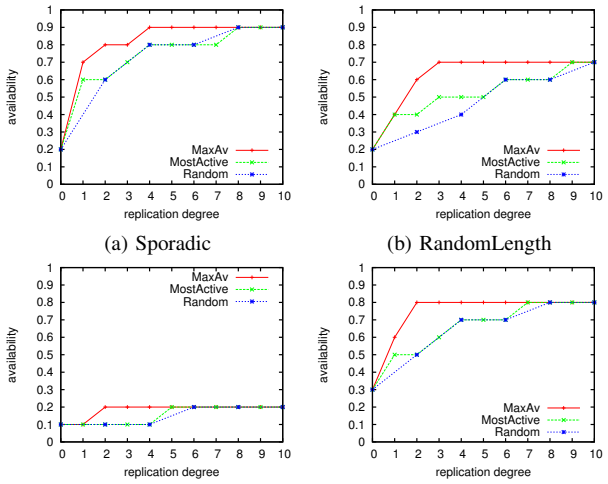
[22] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman, "Lockr: better privacy for social networks," in *Proc. of the CoNEXT*, 2009.

[23] 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.

[24] M. M. Lucas and N. Borisov, "Flybynight: mitigating the privacy risks of social networking," in *Proc. of the WPES*, 2008.

Appendix

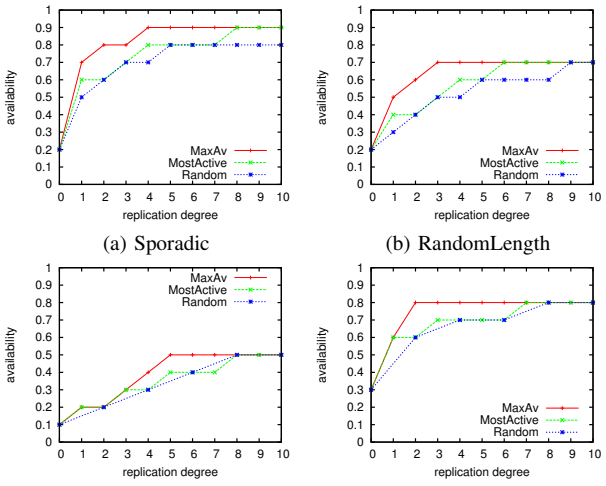
Raw plots



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

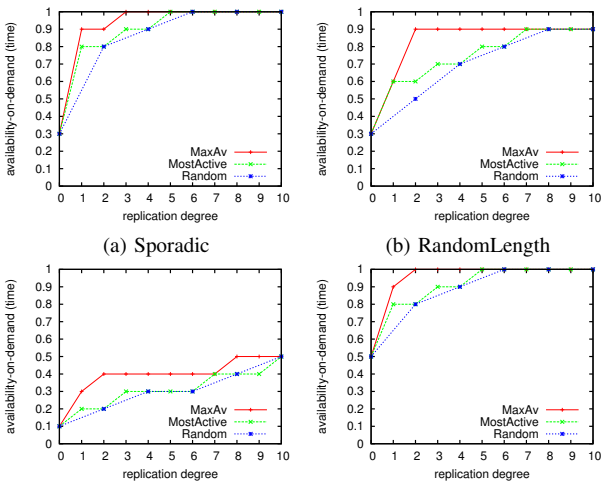
Fig. 17: Facebook-ConRep: Availability



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

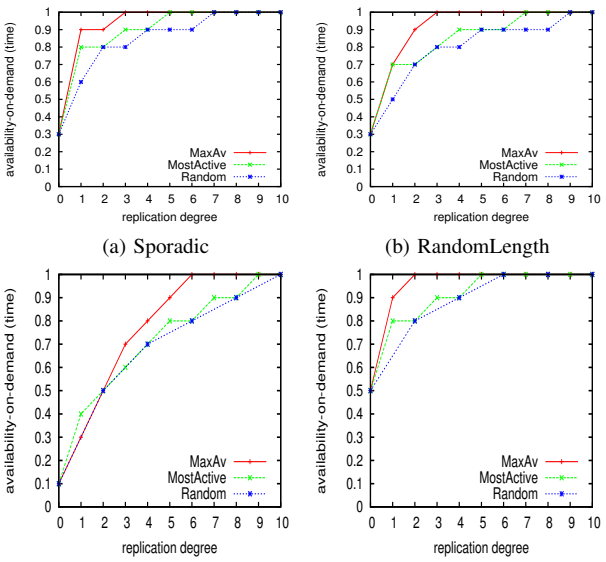
Fig. 18: Facebook-UnconRep: Availability



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

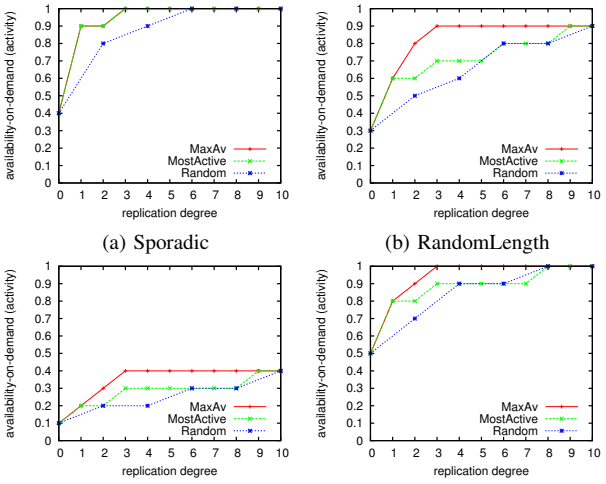
Fig. 19: Facebook-ConRep: Availability-on-Demand w.r.t Time



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

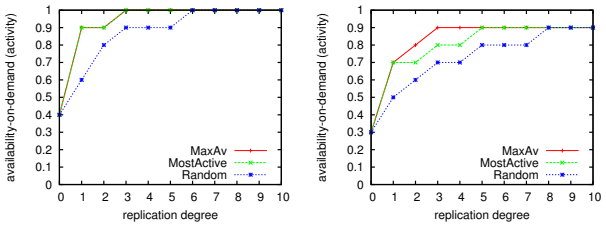
Fig. 20: Facebook-UnconRep: Availability-on-Demand w.r.t Time



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

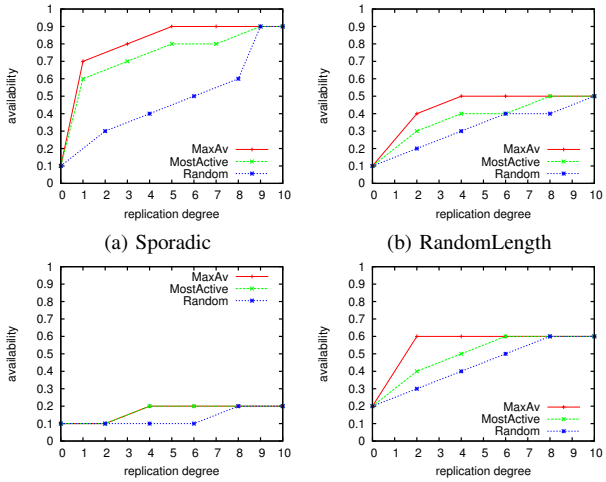
Fig. 21: Facebook-ConRep: Availability-on-Demand w.r.t Activity



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

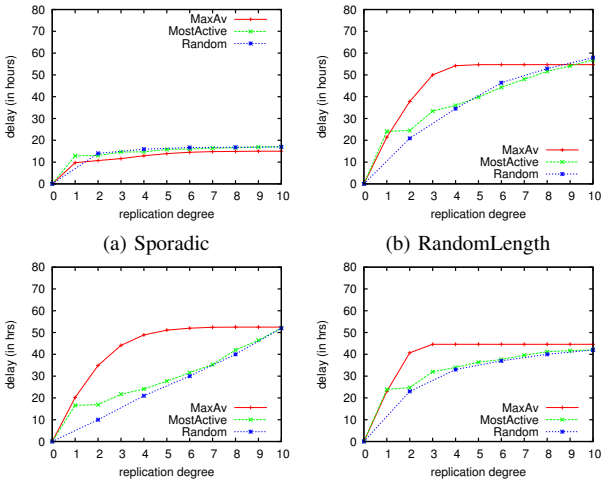
Fig. 22: Facebook-UnconRep: Availability-on-Demand w.r.t Activity



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

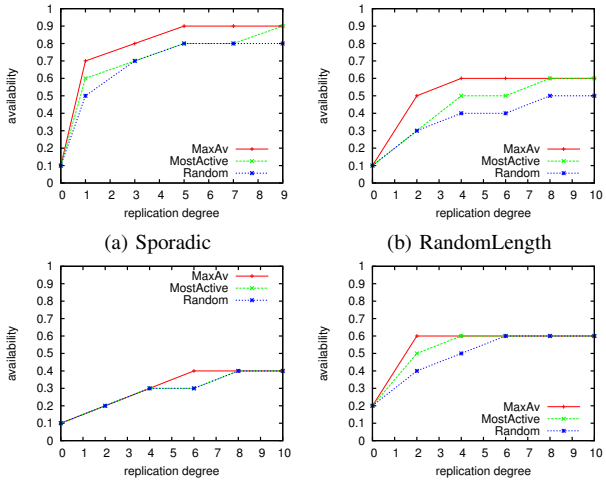
Fig. 24: Twitter-ConRep: Availability



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

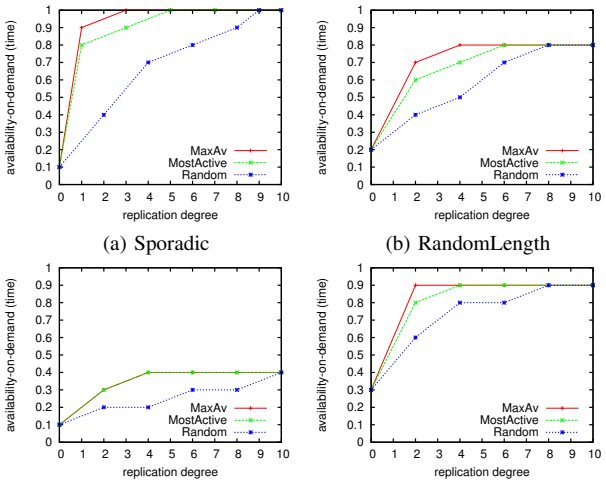
Fig. 23: Facebook-ConRep: Update Propagation Delay



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

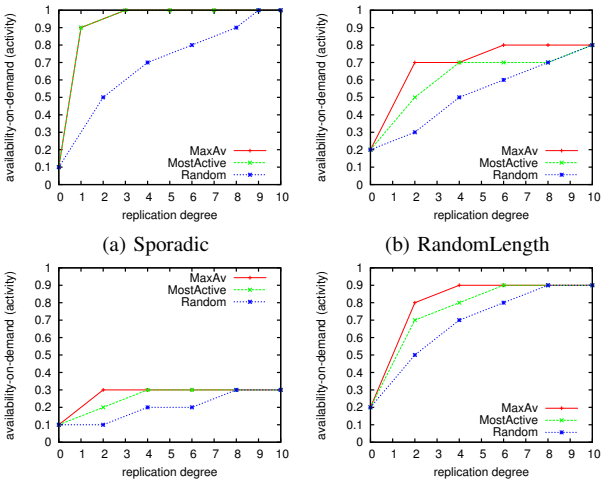
Fig. 25: Twitter-UnconRep: Availability



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

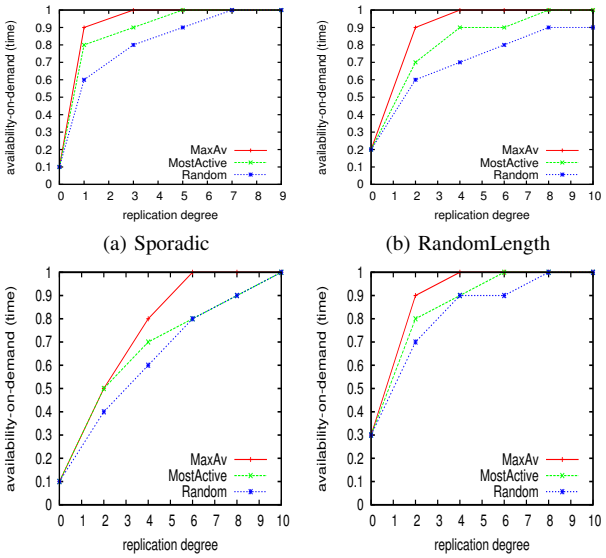
Fig. 26: Twitter-ConRep: Availability-on-Demand w.r.t Time



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

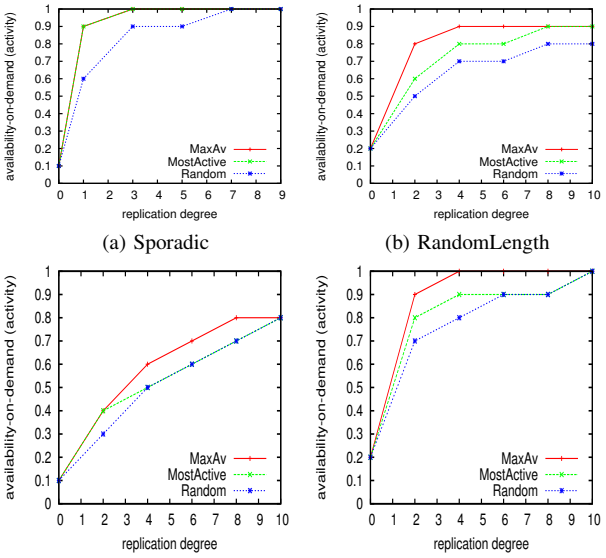
Fig. 28: Twitter-ConRep: Availability-on-Demand w.r.t Activity



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

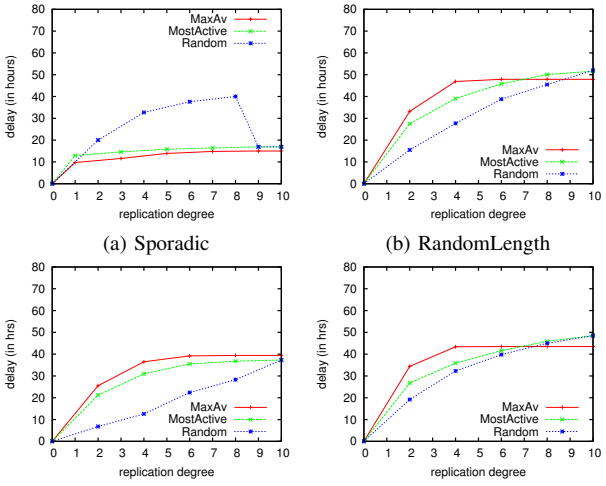
Fig. 27: Twitter-UnconRep: Availability-on-Demand w.r.t Time



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 29: Twitter-UnconRep: Availability-on-Demand w.r.t Activity



(a) Sporadic (b) RandomLength

(c) FixedLength (2hours) (d) FixedLength (8hours)

Fig. 30: Twitter-ConRep: Update Propagation Delay