Predicting the availability of users' devices in decentralized online social networks

Andrea De Salve\textsuperscript{1,2} | Barbara Guidi\textsuperscript{2} | Paolo Mori\textsuperscript{1}

\textsuperscript{1}IIT-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy
\textsuperscript{2}Department of Computer Science, University of Pisa, Largo B. Pontecorvo, 56127 Pisa, Italy

Correspondence
Andrea De Salve, IIT-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy and Department of Computer Science, University of Pisa, Largo B. Pontecorvo, 56127 Pisa, Italy.
Email: desalve@di.unipi.it

Summary
The understanding of the user temporal behavior is a crucial aspect for all those systems that rely on user resources for daily operations, such as decentralized online social networks (DOSNs). Indeed, DOSNs exploit the devices of their users to take on and share the tasks needed to provide services such as storing the published data. In the last years, the increasing popularity of DOSNs has changed the way of how people interact with each other by enabling users to connect to these services at any time by using their personal devices (such as notebooks or smartphones). As a result, the availability of data in these systems is strongly affected (or reflected) by the temporal behavior of their users in terms of connections to DOSNs. In this paper, we propose the use of linear predictors to address the problem of the availability of user devices and, hence, data in DOSNs. To validate the proposed approaches, we evaluated their performance conducting a set of simulations exploiting a dataset of temporal information concerning the connections to Facebook collected from a set of users.

KEYWORDS
availability prediction, decentralized online social networks, user behavior

1 | INTRODUCTION

In the last years, online social networks (OSNs) have proliferated by attracting millions of users that share a large number of contents with each other. According to the International Telecommunication Union (ITU), OSNs have reached a global penetration rate of 31%. The evolution of mobile technology (such as smartphones and tablets) is the major driver that makes social networking the most popular online activity, which captures 30% of the time spent by users each day on the Internet, i.e., about 1 hour and 50 minutes.

The most popular OSNs are based on a centralized architecture where the service provider takes control over users' information. However, the centralized structure has several relevant drawbacks. For instance, during the last years, users' privacy in OSNs becomes an ever-increasing concern because private information of the users is exploited by the centralized OSN providers for several purposes. Privacy and system scalability are two of the main issues that have led to the decentralization of the social networks.

A decentralized online social network (DOSN) is an OSN implemented in a decentralized way (such as peer-to-peer architecture). For instance, Diaspora\textsuperscript{3} and Friendica\textsuperscript{3} are two of the currently available DOSNs that attract more than 669,000 users, and they are based on a network of independent, federated servers that are managed by the users. RetroShare\textsuperscript{4} is another available DOSN that allows users to create friend-to-friend connection for sharing data. In DOSNs, there is no centralized authority since users' devices are exploited to store and manage users' data. For this reason, the decentralization of OSNs requires efficient solutions for dealing with data availability arising when users' devices disconnect from the system. Indeed, the problem of data availability occurs because data of a user should be always accessible even when the owner of the data is offline. Replication is the most widely used technique to maximize data availability, and it consists of storing copies of the same data on several devices and the peers where data are allocated are named replica peers. As a result, data will be available with a certain probability that depends on the strategy with which the replicas were chosen. Indeed, the criteria that are considered when replica peers are selected heavily affect the availability of data.\textsuperscript{4}

\textsuperscript{2}GWI Social report: http://insight.globalwebindex.net/social
\textsuperscript{3}http://diasporafoundation.org/
\textsuperscript{4}http://friendica.com/
\textsuperscript{5}http://retroshare.net/index.html
A first proposal was to randomly select the peers on which the data are placed. However, replicating data on different random users’ devices is not enough to ensure higher data availability because peers participating in the network are heterogeneous in terms of demands and online behavior. To take into account the decentralized nature of social networks, new properties related to the social information (such as friendship type, common interests, closeness, trustworthiness, and tie strength) can be considered during the design of replication strategies.

In recent years, several replication strategies have been proposed, which consider friendships’ information or trustworthiness. In addition, increased awareness about the importance of temporal information has led scientists to investigate new replication strategies that consider general temporal information (such as average session length or online-offline correlation). While several studies leverage the temporal properties of DOSNs’ users in terms of session length or interactions frequency, the availability patterns of users in DOSNs still remain unexplored from replica selection strategies. In general, availability patterns of DOSNs’ users are crucial to better design the data replication strategies of the DOSN infrastructure.

Nowadays, the wide adoption of OSNs enables to collect a huge amount of availability records containing temporal users’ information, which can be used to define predictive tools that envisage the availability of the users’ devices in DOSNs. We believe that the design of such tools will help DOSNs (and decentralized systems in general) in designing new replication strategies that exploit the availability pattern of users.

The focus of this paper is to investigate whether availability patterns of users in OSNs can be predicted by using a linear predictor. The proposed predictor is intended to support decisions of the underlying mechanisms of the DOSNs (such as information allocation/diffusion). For these reasons, we are interested in building a flexible linear predictor that allows to estimate the future availability of a user (i.e., online or offline) by taking into account only the availability history of the user. In order to provide a suitable granularity level and flexibility that meet the specific features of a DOSN, we formally define a linear predictor that depends on the following parameters: (i) the size of the period of time to predict and (ii) the size of the elapsed period of time exploited for the prediction. Since a linear predictor returns the probability that each user will be online at the specified instant of time in the future, we define different selection strategies that help in choosing users more likely to be online in a time interval.

The remainder of this paper is organized as follows. In Section 2, we describe in more detail the reference scenario and the main motivations of our work. We discuss in Section 3 the fundamental background and current status of availability prediction in DOSNs. We formally define the linear predictor and its extension based on intervals, respectively, in Sections 4 and 6, whereas in Section 5, we describe how the proposed predictor is applied to a real use case. In Section 7, we present the methodology used to conduct the experiments and the real Facebook dataset used for the evaluation. Finally, in Section 8, we report the conclusions and discuss future works.

2 | REFERENCE SCENARIO AND MOTIVATIONS

In this section, we introduce in more detail the scenario of our work. Predicting availability of users is one of the main problems in DOSNs. Indeed, such system grounds on a P2P overlay where devices, corresponding to users, connect to each other in order to share storage, tasks, and resources. A trend of current DOSNs consists of modeling a type of P2P network in which users only make direct connections with people they know, i.e., taking into account the friendship relations among them. The network topology resulting from this structure is generally known as friend-to-friend network (F2F). An F2F can be formally represented by using a well-known social network model known as Ego Network, which consists of a user (the ego), its direct friends (the alters), and the social ties occurring between them. In addition, we assume a one-to-one mapping between the users of the OSN and the nodes of the DOSN, and we use the name nodes, devices, or users interchangeably in the rest of this paper. The ego network model better reflects the local and limited knowledge that each user of the DOSN knows. Indeed, a typical DOSNs’ user is aware of its friends, which are included in its ego network, and the friendships between them. An example of Ego Network for the ego node E is shown in Figure 1. As a result,
proper techniques must be introduced in order to ensure that data of the ego users will be available on a subset of their alters. Indeed, the replication of the user’s contents on a set of alters’ peers present in its ego network is one of the most common techniques used in DOSNs to ensure data availability. For these reasons, we consider the ego network of each node as the general reference DOSN model where we apply the linear predictor. In order to ensure that data still remain available in the system, every time a replica node disconnects from the DOSN framework, a new node has to be selected and data have to be copied on it. We want to stress that this overhead is required each time a replica node disconnects from the system. As result, if not properly selected, replica nodes seriously affect the scalability and the performance of the system because the number of connected users could change very often and disconnections of replica nodes impact the data availability.

In our previous work, we showed that the structural properties of the ego network (i.e., friendship relations) are not enough to ensure higher availability and we proposed a new approach to manage the problem of data availability, which considers both structural and temporal properties. For these purposes, we used straightforward temporal properties such as the average session time, which is helpful to choose nodes that are online more than others. However, the temporal properties considered are very simple, and they do not provide a real approximation of the temporal behavior of users. We believe that exploiting availability patterns of users would ensure a higher level of availability. For these reasons, in this manuscript, we use a linear predictor to model past user behavior of users and to predict availability status of the user during a given future time interval.

3 | BACKGROUND AND RELATED WORKS

In recent years, DOSNs have been proposed in order to overcome the limitations in terms of privacy of the current centralized OSNs. As explained in the work of Guidi et al., the decentralization of OSNs requires to face several challenges. One of the main important challenges is guaranteeing the availability of data even when the owner is not online. In this section, we discuss about the related work proposed for the following two aspects: current DOSN proposals and the data availability problem in DOSNs.

3.1 | Decentralized online social networks

Several current DOSNs are built on top of a P2P network, which has been considered a suitable architecture to implement a DOSN by scientists. Diaspora, with about 669 000 users, is one of the few successful DOSN services currently active and deployed. Diaspora relies on an infrastructure that is based on distributed servers (named pod). Data can be encrypted and everyone can set up a server for Diaspora so that availability is ensured by many distributed servers.

LifeSocial is a DOSN where users employ public-private key pairs to encrypt profile data that are securely stored in a FreePastry-based DHT. User information is stored in the form of distributed linked lists in a DHT and is accessible from various plugin-based applications while enforcing symmetric PKI to ensure user-controlled privacy and access. However, the data of a user are isolated from other users, and users access them individually, incurring significant system overhead.

PeerSoN is a distributed infrastructure for OSN service focused on privacy issues. It is a two-tier system architecture in which the first tier is used for the lookup service implemented by a DHT, whereas the second tier consists of nodes representing users, which is used for the communication between peers and the exchange of social data, such as user’s posts. All data are encrypted and stored into the DHT. The second tier allows for opportunistic and delay-tolerant networking.

Safebook addresses privacy issues in DOSNs by storing encrypted profile content in a P2P storage infrastructure. In particular, it uses a trusted identification service, which provides each peer an unambiguous identifier, a P2P layer for lookup of data, and a social overlay implemented by Matryoshkas, concentric rings of peers built around each peer to provide trusted data storage, profile data retrieval, and obscure communication through indirection.

DiDuSoNet is a P2P DOSN based on the trust concept. Users have full access control on their data because the system follows Dunbar’s concept and stores the user’s data only on trusted friends. The system exploits trust relationships between users for providing a set of important social services such as information diffusion and data availability.

Table 1 summarizes the characteristics of existing DOSNs in terms of replication strategy, storage service, and encryption schemes. The majority of the DOSNs rely on a storage service that involves all the nodes of the users, such as PeerSoN and LifeSocial. Diaspora is considered a super-peer.

<table>
<thead>
<tr>
<th>DOSN</th>
<th>Replication Strategy</th>
<th>Storage Service</th>
<th>Encryption Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diaspora</td>
<td>Pods trusted by user</td>
<td>Federated Servers</td>
<td>RSA + Symmetric Keys (AS)</td>
</tr>
<tr>
<td>LifeSocial</td>
<td>Random peers</td>
<td>DHT</td>
<td>RSA + Symmetric Keys (AS)</td>
</tr>
<tr>
<td>PeerSoN</td>
<td>Random peers</td>
<td>DHT</td>
<td>PKI</td>
</tr>
<tr>
<td>Safebook</td>
<td>Peers of the Matryoshkas</td>
<td>DHT</td>
<td>RSA + Symmetric Keys (AS)</td>
</tr>
<tr>
<td>DiDuSoNet</td>
<td>Peers of the Dunbar circles</td>
<td>Trusted Nodes</td>
<td>No Encryption</td>
</tr>
</tbody>
</table>
based architecture because storage service involves only nodes of the federated networks. Instead, in Safebook and DiDuSoNet, the storage service exploits only a subset of the user’s friends. All the previous DOSNs ensure the privacy of the contents by exploiting a combination of symmetric and asymmetric encryption schemes.

3.2 | Data availability in DOSNs

Almost all the existing DOSNs rely on servers or other external storage services to guarantee data availability, and they do not directly managed this problem. Only during the last years, researchers have been focused on studying the problem of data availability in DOSNs. The requirements to guarantee data availability in DOSNs are different from those of the classical distributed storage. The main object that has to be stored is the profile of a user. Usually, user’s profiles contain several parts that are frequently updated but are required only by a small subset of users (the friends of the profile’s owner). For this reason, many techniques that are generally used in P2P distributed storage environments are not suitable in DOSNs. Replication is considered one of the most suitable approaches to manage the data availability issue. Different strategies exist to achieve data availability through replication.21 The easiest one is the random replica nodes selection, which can lead to a high number of replicas to guarantee a high level of availability. Instead, a current trend is the choice of friend nodes to store social data because they are the principal recipients of the user’s content.

Several recent approaches use user behavior as a guideline to evaluate good replica nodes. MY3 exploits user behavior to build an online time graph that is used by the selection strategies to elect replica nodes.9 A similar work is proposed10 where a combination of both temporal and structural properties is used to elect replica nodes. In the work of Schiöberg et al.7 and Sharma et al.22 a set of replication strategies, which exploits both online behavior and relationships between users, is proposed. Moreover, criticisms about choosing the best subset of friends are proposed in other works.21,22 These criticisms are related to both the localization and the lack of friends. However, the main problem is related to the replication strategies. In detail, a replication technique should elect a proper number of replica nodes which both respect particular properties, such as trust between users, and ensure a good level of data availability. The first issue could be faced by considering structural properties of the graph, whereas the second issue could be faced by choosing replica nodes which will be online for a long period. To conclude, user behavior is the main feature to guarantee a high level of data availability and it is considered a necessary condition but not sufficient because a good replication strategy needs to combine online behavior with other features such as social characteristics of the social graph.

3.3 | Availability modeling and prediction

Availability prediction in distributed systems has been largely studied in the work of Bhagwan et al.,23 where they have shown that the availability of 50 hosts of the Overnet P2P system mainly depends on the time of day and it is independent on the availability of other hosts. Stutzbach and Rejaie24 characterize dynamics of the peers participating in different P2P systems, and they demonstrate that both inter-arrival times and sessions-length are better described by a Weibull distribution. Kermarrec et al.14 show that anti-correlation between availability patterns and unavailability periods of two peers increases the performance of the distributed storage. Mickens and Noble22 propose a set of predictors to estimate future availability of users. The first predictor counts the current online status of a peer, the second predictor exploits a De Bruijn graph over k bits to identify availability pattern of users, and the third predictor is a linear predictor without weights. In addition, the authors propose a hybrid predictor that dynamically selects the best predictor for a lookahead period. The proposed approach has a compelling adaptation strategy but it exploits a set of very simple predictors (mainly based on pattern and status counters). In contrast to the approach proposed by the authors, our solution exploits only linear predictors with different weight configurations.

In the work of Blond et al.,26 a new approach to find good partners based on epidemic protocols is proposed. The authors introduce two formal problems: the disconnection matching and the presence matching patterns. Furthermore, authors provide a simple predictor that, by using only 7 days of history, it is able to predict the online periods for the next week. However, the authors mainly focused on the evaluation of the matching epidemic protocol and the linear predictor does not exploit weights.

In the work of Dell’Amico et al.,27 a probabilistic logistic regression classifier is proposed. The authors exploit five different features, which represent global and individual characteristics of user behavior, inside the model. The model is evaluated on three real datasets, two of them publicly available. Furthermore, the paper proposes a comparison between the model and the method proposed by Mickens and Noble.25 Compared with this work, our approach exploits a linear predictor that takes into account the availability information of users that belong to the ego network because in a decentralized environment it is not easy to compute complex global features. Furthermore, we investigated the efficiency of our predictor by using several weights strategies, which permit us to model specific characteristics of users’ pattern. In addition, we investigate the feasibility of the proposed predictor by considering availability information collected from a real OSN scenario.

4 | USING LINEAR PREDICTOR TO ESTIMATE USER AVAILABILITY

Linear prediction has had a significant impact in the field of signal processing and statistical time series analysis.28 Formally, a linear predictor is defined as a linear combination \( f(x^t) \) constructed from a set of \( k \) terms \( x^t_1, \ldots, x^t_k \) by multiplying each term \( x^t_i \) with the corresponding weight \( \beta_i \):

\[
f(x^t) = \beta_1 x^t_1 + \beta_2 x^t_2 + \cdots + \beta_k x^t_k.
\]


<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u )</td>
<td>identifier of a generic user</td>
</tr>
<tr>
<td>( d )</td>
<td>an arbitrary day</td>
</tr>
<tr>
<td>( t_i )</td>
<td>a discrete time of a day represented by positive integers ( i = 1, 2, \ldots, T )</td>
</tr>
<tr>
<td>( T )</td>
<td>the number of time instants in a day</td>
</tr>
<tr>
<td>( \Delta )</td>
<td>the difference (in seconds) between two consecutive instants of times</td>
</tr>
<tr>
<td>( x )</td>
<td>number of past days considered as Availability History Size</td>
</tr>
<tr>
<td>( y )</td>
<td>number of future time instants to predict (Prediction Window)</td>
</tr>
<tr>
<td>( G^y_u \in {0,1}^k )</td>
<td>availability status (online=1, offline=0) of user ( u ) for a day ( d )</td>
</tr>
<tr>
<td>( M_{ij}(d, x, y) \in {0,1}^{xy} )</td>
<td>Availability History Matrix of the user ( u ) considered at a time ( t_i ) of the day ( d ) for a Prediction Window ( y ) and Availability History Size ( x )</td>
</tr>
<tr>
<td>( w \in \mathbb{R}^u )</td>
<td>vector of probabilities returned by the predictor for user ( u ) at time ( t_i )</td>
</tr>
<tr>
<td>( \theta \in [0,1] )</td>
<td>threshold value used in the binary discretization</td>
</tr>
<tr>
<td>( f )</td>
<td>stencil function</td>
</tr>
<tr>
<td>( h )</td>
<td>size of the stencil</td>
</tr>
<tr>
<td>( s )</td>
<td>1-dimensional grid array of size ( 2h + 1 ) that defines the offset ( s[j] ) and the weight ( s_j ) of the stencil</td>
</tr>
</tbody>
</table>

In particular, weights \( \beta_i \) for \( i = 1, \ldots, k \) are named coefficients, and they are used to fine-tune the impact of each independent variable \( x^i \) in predicting the outcome of a dependent variable \( f(x^i) \). For the purposes of data availability, linear predictors can be used to calculate the probability that a user \( u \) is online/offline in a given future period \( t \) by taking into account the past availability status of the same user \( u \) in \( k \) different time instants.

### 4.1 Our approach: a linear predictor based on time intervals

In our approach, we assume a discrete time model, where time \( t_i \) of a day \( d \) is represented by positive integers \( i = 1, 2, \ldots, T \), where the number of time instants \( T \) in a day depends on the sampling frequency and it is chosen to match the need of the application. The difference between two consecutive instants of time \( t_i – t_{i-1} \) of the same day (time granularity) is represented by \( \Delta \). Table 2 summarizes the notation used for the definition of the predictors. The online presence of a user \( u \) at a certain time instant \( t_i \) of a day \( d \) is represented by the term \( G^u_d[i] \), which can have a binary value, i.e., 1 or 0, depending on the presence or absence of the user \( u \), respectively. In particular, given an arbitrary time instant \( t_i \) of a day \( d \), we want to predict whether the user \( u \) will be online (or offline) in the future time interval \([t_{i+1}, t_{i+y}]\) of day \( d \) by taking into account the online/offline status of \( u \) at the same instants of time \([t_{i+1}, t_{i+y}]\) of the previous days \([d-1, d-2, \ldots, d-x]\). The parameter \( y \) is named Prediction Window, and it indicates the future time instants in which we want to predict the online presence of a user. Instead, we refer to the parameter \( x \) as Availability History Size since it indicates the past availability information of users (in terms of the number of days prior to the time \( d \)) considered by the predictor. For the sake of clarity, we represent availability history of a user \( u \) at a time \( t_i \) of the day \( d \) through a binary Availability History Matrix \( M_{ij}(d, x, y) \in \{0,1\}^{xy} \), where \( y \) and \( x \) are respectively the Prediction Window and the Availability History Size. The Availability History Matrix \( M_{ij}(d, x, y) \) is defined as follows:

\[
M_{ij}(d, x, y) = \begin{bmatrix}
G^u_d[d-1+1] & G^u_d[d-1+2] & \cdots & G^u_d[d-1+y] \\
G^u_d[d+1+1] & G^u_d[d+1+2] & \cdots & G^u_d[d+1+y] \\
\vdots & \vdots & \ddots & \vdots \\
G^u_d[2h+1+1] & G^u_d[2h+1+2] & \cdots & G^u_d[2h+1+y]
\end{bmatrix},
\]

(2)

where each row at index \( j \) of the Availability History Matrix, i.e., \( M_{ij}(d, x, y) \), specifies the online presence (0 or 1) of a user \( u \) at the time instants \( t_{i+1}, t_{i+2}, \ldots, t_{i+y} \) of the past day \( j = d-1, d-2, \ldots, d-x \). To keep down the complexity of the linear predictor, we exploit only the availability information obtained by the Availability History Matrix to make predictions about the user’s availability.

The rationale behind our approach is that the availability status of a user \( u \) at a future time \( t_{i+1} \) of the day \( d \) could depend on the availability status of \( u \) at the same time \( t_{i+1} \) of the previous days \( d-1, d-2, \ldots, d-x \). For example, as shown in other works,29,30 users of the OSNs seem to connect to the service with a periodic trend. In particular, groups of students of the same school connect to the OSN outside the hours of lessons and expose a periodical time pattern in their connections.29 This suggests the existence of a periodic pattern where each user is connected at similar times each day. We believe that such connection patterns of users can be exploited to achieve accurate predictions about the availability of users. For these reasons, we define a predictor where the probability that \( u \) will be online/offline at the time instant \( t_{i+1} \) is computed by considering the availability status of \( u \) at time instant \( t_{i+1} \) of the previous days \( d-1, d-2, \ldots, d-x \). The linear predictor accepts as input parameter the current instant of time \( t_i \) of the day \( d \) where to start the prediction, a target user \( u \), the Prediction Window \( y \), the Availability History Size \( x \) of user \( u \), and the corresponding Availability History Matrix \( M_{ij}(d, x, y) \). In addition, we assume that previous days \( d-1, d-2, \ldots, d-x \) could have a different impact (or relevance).
on the availability of $u$. In order to improve the expressiveness of our approach, we introduce also the coefficient vector $w \in \mathbb{R}^x$ that specifies the weights reflecting the importance of each of the previous $x$ days. The coefficient vector can be used in the fitting process to indicate how much each day contributes to the prediction of the availability status. Once these components are defined, the linear predictor returns a vector $P_u \in \mathbb{R}^y$ where the $j$-th element specifies the probability $P_u[j]$ with which user $u$ is online/offline at the time instant $t_{ij}$ (where $0 < j \leq y$) of the Prediction Window.

Each element $j$ of the vector of probabilities is computed as follows:

$$P_u[j] = \frac{\sum_{k=1}^{x} w_k \cdot M_k(i, x, y) [k][j]}{\sum_{k=1}^{x} w_k} \quad \forall j = 1, \ldots, y. \quad (3)$$

In particular, we apply a linear combination between the coefficient vector $w$ and the online status of the user $u$ at time $t_{ij}$ of the past days $d - 1, d - 2, \ldots, d - x$. To be able to compare probability measure, we performed a normalization by dividing it for the sum of the weights $w$, obtaining a normalized value between 0 and 1. In the same way, the general computation of the probability vector $P_u$ is shown by Formula (4) as follows:

$$P_u = \frac{1}{\sum_{k=1}^{x} w_k} \begin{pmatrix} w_1 & w_2 & \cdots & w_x \end{pmatrix} \times \begin{pmatrix} G_{d-1}[i+1] & \cdots & G_{d-1}[i+y] \\ G_{d-2}[i+1] & \cdots & G_{d-2}[i+y] \\ \vdots & \ddots & \vdots \\ G_{d-x}[i+1] & \cdots & G_{d-x}[i+y] \end{pmatrix}. \quad (4)$$

### 4.2 Defining weights of relevance

The investigation of the criteria used to assign weights of the coefficient vector $w$ is crucial to achieve higher accuracy of the prediction because it allows to express the different importance given to the presence of a user in past days. Experimental results (such as those reported in the work of Bhagwan et al.\textsuperscript{23}) have shown that users of a specific decentralized system have their availability pattern, which is different from that of another system. The weights’ values allow to adapt the prediction to the distinct conditions that affect the availability status of individual users. Therefore, the weights of the prediction model can differ significantly according to the specific characteristics of the users. Let $x$ be the number of past days considered as Availability History Size, ordered in chronological order, from the most recent $j = 1$ to the oldest $j = x$. Hence, we propose the following strategies for defining weights of the coefficient vector $w$.

- $w^x$ Each day $j$ has the same importance in predicting the user’s availability, and each weight $w_j$ of the coefficient vector is equal to 1.
- $w^w$ Each day $j$ takes a weight $w_j = (x - j + 1)$ that is proportional to its age.
- $w^c$ Each day $j$ has a weight $w_j = 1/2^j$ that decreases exponentially with respect to its age.

Figure 2B shows the weights assigned by coefficient vectors to the days preceding the prediction day $d$. The coefficient vector $w^x$ is meant for users who behave according to a regular availability pattern over time, i.e., users who are connected to the OSN at about the same time every day. For example, let us consider the case of users who have fixed working hours: during the working days, these users can connect to the OSN only outside their working hours. In such a case, it is very likely that user $u$ behaves similarly at the same time instant $t$ of distinct days, and the coefficient vector $w^x$ fits well this situation. However, in some cases, it is possible that users do not behave regularly because their availability patterns are mostly affected by the occurrence of new recent events. As for example, college students may increase or decrease the time spent on OSNs over the days depending on both their workload and the academic calendars. In such a case, we expected that very old information on the availability status of a user is outdated and cannot be exploited for prediction. Instead, the availability status of users in the days immediately preceding $d$ may have more influence on the current status of users. As a result, the coefficient vectors $w^w$ and $w^c$ are intended for users whose availability patterns depend

---

**FIGURE 2** Figure (A) shows a graphical representation of the computation performed by the predictor, whereas Figure (B) shows the weights given to the availability status of the days preceding $d$ by each coefficient vector.
mainly on the past availability status over a short period of time. In particular, vector \( w^0 \) gives to past availability status importance, which decreases linearly with its age, whereas vector \( w^1 \) assigns to past availability status a weight value that decreases exponentially with its age. With respect to \( w^0 \), the coefficient vector \( w^1 \) assigns greater importance to the recent availability status of a user and lesser importance to old availability status. As a result, the proposed coefficient vectors implement different adaptation techniques in order to react to changing conditions in the users’ availability patterns. Let us assume that future availability status of user \( u \) is predicted by considering the following past availability status \([1, 1, 1, 0, 0]\) arranged in chronological order. Then, the linear predictors with coefficient vectors \( w^0, w^1, \) and \( w^2 \) will consider user \( u \) online with high probability (i.e., about 0.6), with medium probability (i.e., about 0.4), and with low probability (i.e., about 0.2), respectively.

### 5 EXAMPLE

We show the usage of the linear predictor defined above by considering a concrete example where we assume that, at time \( t_i \) of day \( d \), a user \( u \) wants to predict the availability status of its ego network that consists of three friends (Alice, Bob, and Charlie) in the next four time instants \( t_i, t_{i+1}, t_{i+2}, t_{i+3} \), (i.e., \( y = 4 \)) by exploiting the availability history of users in the previous five days (i.e., \( x = 5 \)). We assume that the availability status of each user is observed periodically (every \( \Delta = 10 \) minutes) and the Availability History Matrix of the users are defined as follows:

\[
M'_A(d, 5, 3) = \begin{bmatrix}
G^A_{i-1}[i+1] & G^A_{i-1}[i+2] & G^A_{i-1}[i+3] & G^A_{i-1}[i+4] \\
G^A_{i-2}[i+1] & G^A_{i-2}[i+2] & G^A_{i-2}[i+3] & G^A_{i-2}[i+4] \\
G^A_{i-3}[i+1] & G^A_{i-3}[i+2] & G^A_{i-3}[i+3] & G^A_{i-3}[i+4] \\
G^A_{i-4}[i+1] & G^A_{i-4}[i+2] & G^A_{i-4}[i+3] & G^A_{i-4}[i+4] \\
G^A_{i-5}[i+1] & G^A_{i-5}[i+2] & G^A_{i-5}[i+3] & G^A_{i-5}[i+4]
\end{bmatrix}
\]

\[
M'_B(d, 5, 3) = \begin{bmatrix}
G^B_{i-1}[i+1] & G^B_{i-1}[i+2] & G^B_{i-1}[i+3] & G^B_{i-1}[i+4] \\
G^B_{i-2}[i+1] & G^B_{i-2}[i+2] & G^B_{i-2}[i+3] & G^B_{i-2}[i+4] \\
G^B_{i-3}[i+1] & G^B_{i-3}[i+2] & G^B_{i-3}[i+3] & G^B_{i-3}[i+4] \\
G^B_{i-4}[i+1] & G^B_{i-4}[i+2] & G^B_{i-4}[i+3] & G^B_{i-4}[i+4] \\
G^B_{i-5}[i+1] & G^B_{i-5}[i+2] & G^B_{i-5}[i+3] & G^B_{i-5}[i+4]
\end{bmatrix}
\]

\[
M'_C(d, 5, 3) = \begin{bmatrix}
G^C_{i-1}[i+1] & G^C_{i-1}[i+2] & G^C_{i-1}[i+3] & G^C_{i-1}[i+4] \\
G^C_{i-2}[i+1] & G^C_{i-2}[i+2] & G^C_{i-2}[i+3] & G^C_{i-2}[i+4] \\
G^C_{i-3}[i+1] & G^C_{i-3}[i+2] & G^C_{i-3}[i+3] & G^C_{i-3}[i+4] \\
G^C_{i-4}[i+1] & G^C_{i-4}[i+2] & G^C_{i-4}[i+3] & G^C_{i-4}[i+4] \\
G^C_{i-5}[i+1] & G^C_{i-5}[i+2] & G^C_{i-5}[i+3] & G^C_{i-5}[i+4]
\end{bmatrix}
\]

Each row \( j \) of the Availability History Matrix contains the availability status (online=1 or offline=0) of a user in the time intervals \([t_i, t_{i+1}]\) of the days \([d−1, d−x]\). In particular, we can graphically represent the availability history of a user by means of a graph (see Figure 2A), where, on the x-axis, there are the time instants monitored during a day and, on the y-axis, there are the past days in which the user’s availability was monitored. The availability information used by the predictor to predict the availability status of users is highlighted in red, whereas the time instants highlighted in blue are instants of the time for which we want to predict the availability of users A, B, and C.

In this example, the coefficient vector used to assign weights to each day is chosen according to the strategy \( w^d \) defined in Section 4.2. The vector of probabilities \( P_A, P_B, \) and \( P_C \), resulting from the application of Formula (4), is shown by the following formulas and they are computed by multiplying each column of the Availability History Matrix with the coefficient vector:

\[
P'_A = [0.4 \ 1.0 \ 0.2 \ 0.8]
\]

\[
P'_B = [0.8 \ 0.4 \ 0.4 \ 0.8]
\]

\[
P'_C = [1.0 \ 0.2 \ 0.2 \ 1.0]
\]

Each element \( j = 1, 2, \ldots, y \) of the vector \( P'_u \) represents the probability that user \( u \) will be online at future time instant \( t_{i+j} \). Given the availability vectors returned by the linear predictor, different strategies become available to select users that will be online (with high probability) in the considered time interval \([t_{i+1}, t_{i+4}]\). As for instance, users may be selected so as to maximize the likelihood to have online users by choosing, in each time instant of the interval \([t_{i+1}, t_{i+4}]\), the user who has the highest probability of being online. In the following section, we discuss and propose different strategies that can be adopted to select users.
6 | EXTENDING THE LINEAR PREDICTOR WITH INTERVALS

The predictor defined by Formula (4) calculates the probability \( P_u \) that a generic user \( u \) will be online/offline at time instants \( t_{\infty}, \ldots, t_{\infty} \). As for instance, let us consider the probability vectors \( P_{1s}, P_{2s}, \) and \( P_c \) resulting from the example in Section 5, and we assume that users A and B have been chosen as possible replica nodes because they prove to be more online during the time instants \( t_1, t_2, \ldots, t_{\infty} \) (see Formulas (8), (9), and (10)). In such a case, it is possible that replica nodes A or B will be online at separate time instants and there is no overlapped period of time in which A and B are both online. As a result, the current replica B cannot send the data to the replica A. In particular, we assume that replica node B is online only for the time instants \( t_2, t_3, \) and \( t_{\infty} \) while replica node A is online in the time instant \( t_4 \). In such a case, during the time interval \( t_2, t_{\infty} \), there is no way for the current replica B to send the data to the replica A because there is no online time in which both A and B are online simultaneously. Indeed, user B is already offline when user A connects to the system (i.e., at the time instant \( t_{\infty} \)) and data are no longer available as long as user B reconnects to the system.

A further refinement that we propose is to extend the calculation of the probability \( P_u \) to consider also the overlapping between consecutive time instants. In particular, we define a time window centered around the time instant \( t_i \), which allows to take into account the availability status of user in a time interval centered at time \( t_i \). For this reason, we apply to each element of the availability history \( G_u^d[i] \) a stencil function \( f \) that takes as input parameters the availability history \( G_u^d[i] \) of the instant of time \( t_i \) to consider, and a 1-dimensional grid array \( s \) of size \( 2 \cdot h + 1 \) that determines how the stencil operates on the input value \( G_u^d[i] \). In particular, the 1-dimensional grid array \( s \) is a map \( s \) that defines how the availability status of \( G_u^d[i] \) at time \( t_i \) depends on its neighborhood (i.e., \( G_u^d[i\pm1], G_u^d[i\pm2], \ldots, G_u^d[i\pm h] \)), and it is graphically represented by an array of fixed offsets related to the input position \( t_i \):

\[
s = [-h \ldots -1\, 0\, 1 \ldots +h]
\]

As an example, considering a generic user \( u \), we assume that the following 1-dimensional grid array \( s = [-1, 0, 1] \) of size 3 has been defined. Then, the availability status \( G_u^d[i] \) at time \( t_i \) of a day \( d \) is updated by considering the availability status of \( u \) in the following time instants: (i) at time \( t_i \), i.e., \( G_u^d[i] \); (ii) at the time instant immediately before \( t_i \), i.e., \( G_u^d[i-1] \); and (iii) at the time instant immediately after \( t_i \), i.e., \( G_u^d[i+1] \). The stencil function \( f(G_u^d, t_i, s) \) defines both how to combine the values resulting from the map \( s \) in order to obtain an update value of \( G_u^d[i] \) and the extent to which these values affect the output. For these reasons, we assume that the map \( s \) contains information about the importance of each value and the term \( s_j \) is used to indicate the weight assigned to the value related to \( s[j] \).

The stencil function is defined by Formula (11) and it returns a linear combination of the input values by multiplying each of them by the corresponding weight and adding them to the results.

\[
f(G_u^d, t_i, s) = \sum_{j=1}^{2h+1} s_j \cdot G_u^d[i + s(j)] \sum_{j=1}^{2h+1} s_j
\]  

(11)

The resulting predictor is shown by Formula (12) and it differs with respect to the previous one (shown by Formula (3)) in that it considers a stencil function \( f \) having a 1-dimensional grid array \( s \) of size \( 2h + 1 \).

\[
P_u[j] = \frac{\sum_{k=1}^{2h+1} w_k \cdot f(G_u^d, t_{\infty}, s)}{\sum_{k=1}^{2h+1} w_k} \quad \forall j = 1, \ldots, y \quad s = [-h, \ldots, 0, \ldots, h]
\]  

(12)

We observe that the linear predictor defined above is very simple and it can be easily adopted in distributed and decentralized systems without a particular impact on the complexity and on the performance of the system.

Finally, we observe that the values \( P_u \) resulting from the application of the linear predictor at generic time instant \( t_i \) range between [0,1], and they allow to measure the probability that the user \( u \) will be either online or offline. However, in some cases, it is useful to have a binary predictor that simply indicates the presence (online) or not (offline) of a user \( u \) in a given time instant. Eventually, binary discretization function \( z \) can be applied to the values of \( P_u \) in order to compute binary prediction of the user’s availability (i.e., online or offline), and it consists in defining a threshold value \( \theta \in [0,1] \) to be used as cut-point. In particular, when the probability value \( P_u[j] < \theta \), the user \( u \) is considered offline at the time \( t_{\infty} \); otherwise, the user \( u \) is considered online (as shown by Formula (13)).

\[
z(j) = \begin{cases} 
    \text{offline} & \text{if } P_u[j] < \theta \\
    \text{online} & \text{if } P_u[j] \geq \theta 
\end{cases} \quad \forall j = 1, 2, \ldots, y
\]  

(13)

6.1 | User selection strategy

The availability vectors returned by the linear predictor can be used to drive the selection of users that will be online. Indeed, given the availability vectors of several users, it is necessary to assess which of them ensures the best availability in a future time interval \( [t_1, t_{\infty}] \). For this reason, we defined three selection strategies.
The Random strategy randomly selects a set of users regardless of the outcome of the predictor. As a result, this strategy does not depend on the predictor and selection of users does not introduce any overhead.

The MaxTimeSlot strategy chooses, for each time instant $t_{i_1}, t_{i_2}, \ldots, t_{i_y}$, the user $u$ that has the highest probability of being online at the specific time.

$$\text{argmax}_u P^y_{[j]} \quad \forall j = 1, 2, \ldots, y$$

As a result, this strategy maximizes the probability of choosing a user that will be online at a given time instant, regardless of the number of users.

The MaxSumSlot strategy chooses the user $u$ that has the highest probability of being online by considering all time instances $t_{i_1}, t_{i_2}, \ldots, t_{i_y}$. For this reason, we weight each value $j$ of the probability vector $P^y_u$ with a rank score that determines the total probability that a user $u$ will be online for the whole prediction window. Finally, the user $u$ having the maximum value of the probability vector is selected.

$$\text{argmax}_u P^y_{[j]} \cdot \text{rank}(u) \quad \forall i = 1, 2, \ldots, y \quad \text{rank}(u) = \sum_{k=1}^{y} P^y_{[k]}$$

7 | EXPERIMENTAL METHODOLOGY

With the aim of evaluating the performances in terms of accuracy of our predictors, we have developed a set of simulations based on real availability traces.

7.1 | The dataset

In order to collect the data concerning the availability status of Facebook users, we implemented a Facebook application called SocialCircles, which exploits the Facebook Graph API. Since there is no direct way to obtain the time spent online by users and their friends, we used the chat service to track the online status of Facebook’s users. The SocialCircles! application was released in April 2014, and it had been running until the end of April 2015 when Facebook introduced the new Graph API 2.0, which contained several differences in the way the applications can request data to Facebook and removed the permission to access online presence status of Facebook users. Before the release of the new Facebook Graph API, i.e., from 9 March to 10 April 2015, we sampled the availability status of all the registered users and their friends every 5 minutes for 32 days. In particular, SocialCircle! was able to retrieve the following sets of information:

Friendship the set of friends of each registered user and the friendship relations existing between them.

Online presence the time spent online by registered users and their friends. This was obtained by requesting the online presence permission, which allows the application to monitor the presence status of users. The presence status can assume a limited set of values: 0 if user is offline, 1 if user is in active state, and 2 if user is idle (i.e., the user is online but he has not performed actions for more than 10 minutes).

Using this methodology, we were able to access the temporal status of 204 registered users and of their friends (for a total of 87,191 users). Session information can be easily derived from the availability trace of each user by determining the start of a session (when a user switches from offline to online or idle) or the end of a session (when a user switches from online or idle to offline). From the previous analysis conducted on the behavior of users in OSNs, we expected that users connect to the service frequently and for short periods of time. As a result, we decided to fix the granularity of the dataset to 5 minutes (the smallest possible) because a shorter granularity was not possible for technical reasons related to the Facebook Graph API.

7.2 | Characteristics of the dataset

To achieve a deeper understanding of the collected data, we investigate the users’ temporal behavior in Facebook by presenting a complete analysis of our sample data. Figure 3A shows the total number of online users that accessed the Facebook site during the monitoring period. The plot indicates clearly the presence of a cyclic day/night pattern (confirmed also by other results). In addition, the graph confirms the presence of two peaks of time when most of the users seem to be connected: (i) during lunchtime from 13:00 to 15:00 and (ii) in the evening (from 21:00 to 23:00). During the day, about 20% of users (i.e., about 13,000 users) are connected simultaneously to the OSN. Most of the online users (55%) are connected to Facebook without doing any action (i.e., they are in an idle state) while the other users (about 5,000) actively participate in the OSN. We estimate how often and how long users connect to the OSN by measuring the number and duration of sessions for each user. Figure 3B shows the cumulative distribution function (CDF) of the average number of sessions done by users. The average number of sessions of a user is about 4 sessions per day. The 95% confidence interval (CI) is ±0.03 and indicates that the majority of users expose less than 10 daily sessions.

* https://www.facebook.com/SocialCircles-24471990945196/
Figure 3 shows, for all users, the CDF of both the session length and the elapsed time (or inter-arrival time) between two consecutive user’s sessions, respectively. The average number of minutes that each user spends on Facebook during a session is 40 (95% CI \( \pm 0.05 \)). However, users expose very heterogeneous sessions because the median session value indicates that half of the users’ sessions are shorter than 10 minutes. Indeed, there is a small fraction of users (about 10%) who have session length greater than 1 hour, which affects the total average session length. As shown in Figure 3C, inter-arrival time distribution is very skewed and users exhibit an average inter-arrival time between two consecutive sessions equal to 208 minutes (95% CI \( \pm 0.36 \)). However, we can notice that half of the users have an average inter-arrival time shorter than 55 minutes. The temporal behavior of Facebook users is consistent with those resulting from other studies and confirms that users connect to the OSN frequently and for shorter periods of time.

7.3 Simulation settings and performance measures

In order to assess the performance of the proposed predictor and user selection strategies, we implemented a testbed, which allowed us to perform a set of simulations exploiting our real Facebook dataset. Since the results presented in Section 7.2 show that users have an average session length of about 40 minutes and an average inter-arrival time of about 55 minutes, we decided to predict the availability of a user for short periods of time (i.e., a few hours) in the future because it is very unlikely that users remain connected for long periods of time (e.g., 3 or more hours). Indeed, whenever a selected user disconnects from the system, the predictor can be executed again in order to select a new user who is presumed to be online in the next period of time. For sake of clearness, we observe that the dataset is well suited only for availability prediction over short periods of time while it is not suitable for large-scale prediction due to its limited size. Indeed, the dataset consists of about 5 weeks and is not enough for testing the accuracy of the predictions over several weeks. For these reasons, we are not able to capture monthly, seasonal, or annual variations and we focused on short-term predictions i.e., we predict the availability for the next 30, 60, and 120 minutes. To conduct our experiments, we split the dataset into two distinct temporal parts: (i) the training set, which contains the availability status (online or offline) of users in the first 25 days of the monitoring periods (i.e., from 9 March 2015 to 2 April 2015) and (ii) the test set, which contains the availability status of users in the last 7 days of the monitoring periods (i.e., from 3 April 2015 to 10 April 2015). The training set is used as availability history for predicting future availability status of a user. In order to assess how the history size \( x \) impacts the result of our predictors, we decide to use past availability status concerning the latest 7 days or 14 days, respectively, because they allow to consider daily and weekly patterns that occur on the seven-day weekly cycle. As a result, the Availability History Matrix is built by taking into account the availability status of a user of the training set.

For each user \( u \) belonging to our dataset, we predicted the related availability status at a set of future time instants belonging to the test set and we compared the results of the predictions against the real behavior of users recorded in the test set. In particular, for each day \( d \) of the test set, we
performed the prediction at 12 time instants, starting from midnight (00:00) till 10 PM (22:00) every 2 hours. As a result, for each registered user, we executed 12 predictions for each day \( d \) of the test set for a total number of 84 predictions per user. Since we are interested in finding users of the ego network who will be online more than the others when the ego is offline, we fixed the Prediction Window \( y \), respectively, to (i) 60 minutes (i.e., about the average inter-arrival time), (ii) 30 minutes (i.e., below the average inter-arrival time), and (iii) 2 hours (i.e., above the average inter-arrival time). We recall that, in our dataset, the time granularity \( \Delta = t_i - t_{i+1} \) corresponds to 5 minutes. It is important to note that we do not assume that users selected by the predictor will be available for sure in the period of time \( y \) and, in case the selected user disconnects from the system during this period, the predictor can be executed again in order to select a new user who is presumed to be online for the next \( y \) time instants. In particular, a prediction for a registered user \( u \), executed at time \( t_i \) of the day \( d \), selects the friends of \( u \) and computes for each of these friends \( f \) the probability \( P_f \) that \( f \) will be online at a given future period \( t_{i+1}, \ldots , t_{i+y} \) by taking into account the past availability status of \( f \) related to the last \( x \) days. Then, availability vectors resulting from Formula (12) are used by the user selection strategies to choose which friend \( f \) ensures the highest availability in the future time interval \( [t_{i+1}, t_{i+y}] \).

In order to assess the performance of our predictor, we measured the number of time instants in which the availability status of a user was correctly and incorrectly foreseen by the predictor (i.e., true/false positive or TP/FP). In particular, the prediction is defined as a true positive when the user’s status is correctly identified as online. In contrast, the prediction is a false positive when the status of the user is incorrectly identified as online. Then, we exploited the number of true positive and the false negative predictions to compute the following performance measures.

- **Precision** is defined as \( TP/(TP + FP) \) and it indicates the fraction of prediction correctly identified.
- **False Discovery Rate** is defined as \( FP/(TP + FP) \) and it measures the fraction of errors performed by our predictor.

Table 3 specifies the values of the input parameters used by both the linear predictor and the simulations for the evaluation. We used the Random strategy as a lower bound of the performance of the predictor and to evaluate the impact of the MaxTimeSlot and MaxSumSlot strategies.

### 7.4 Experimental results

In this section, we evaluate the proposed linear predictor by presenting the performance measures obtained from the simulations defined in Section 7.3.

#### 7.4.1 Coefficient vectors

The coefficient vectors \( w^a \), \( w^b \), and \( w^c \) allow the predictor to take into account different situations concerning the past availability status of users in order to correctly predict the presence of a pattern. As a result, each coefficient vector leads to a specific performance value of the predictor, depending on the frequency with which the pattern occurs. Figure 4 shows the performance achieved by the user selection strategies when the linear predictor is configured to use a specific coefficient vector. The results clearly indicate that the Random strategy is the worst user selection strategy and its precision does not depend on the coefficient vectors. Regardless of the different user selection strategies, the coefficient vector \( w^c \) achieves the worst average precision if compared with vectors \( w^a \) and \( w^b \). This suggests that the past availability status information about a user is still valuable for the prediction of future availability status even if they are not recent. Indeed, the coefficient vector \( w^b \) achieved the best average accuracy (about 0.85\( \pm0.29 \)) for MaxTimeSlot and 0.769\( \pm0.36 \) for MaxSumSlot if compared with the coefficient vector \( w^a \) (about 0.82\( \pm0.30 \)) for MaxTimeSlot and 0.767\( \pm0.36 \) for MaxSumSlot.

#### 7.4.2 Past days of history

We analyzed the impact of the number of past days considered for the prediction (history size) on the performance of the linear predictor. Figure 5A shows the average precision of two different linear predictors that exploit the user availability history of the last 7 days (History size \( x = 7 \) days) and of the last 14 days (History size \( x = 14 \) days). The vector of probabilities returned by the predictor is used as an input parameter by each of the selection strategies shown in Section 6.1, which return the users that are presumed to be online in the future time interval.
The results clearly show that the precision of the prediction performed by the random strategy is, on average, less than 10%, regardless of the number of past days considered.

Instead, the MaxTimeSlot strategy outperformed the other strategies by providing correct prediction with an average precision of about 85%. In particular, the users selected using the MaxTimeSlot strategy resulted to be online for the most part of the prediction time, and increasing the number of past days (represented with $x$) considered as history, we did not get significant improvement (or degradation) in the prediction performance. Indeed, Figure 5A indicates that the results of the MaxTimeSlot strategy are quite the same in cases of 7 days and of 14 days of history. This suggests that users of our dataset expose a cyclic online pattern that is repeated weekly. For this reason, in our simulations, the availability history of the user of the last 7 days is enough to predict its future availability status, and a history of 14 days does not introduce any noticeable improvement on the precision of the prediction. The average precision of the MaxSumSlot does not exceed 0.8, and Figure 5A indicates that a slight improvement occurs on the performance of the predictor when the size of history is larger (i.e., equal to 14 days). Indeed, the MaxSumSlot strategy selects the set of users who are presumed to be online for the most part of the future time interval. As a result, this strategy does not take into account the behavior of users in each specific time slot of the future time interval and it is possible that the users selected by using this strategy do not cover the overall future time interval, i.e., there may be some time slots in which none of the selected users is online. The increase in the history size improves the performance of the MaxSumSlot strategy because old availability information of the users allows to better assess the extent to which users will be online in the considered future time interval.

Since Figure 3A suggests that users tend to be connected during specific times of the day, we evaluate the average precision of the linear predictor on different time instants $t_i$ of the day (as specified by Table 3). To this aim, we compute the average precision achieved by the linear predictor when predicting the availability status of users at 12 different time instants equally distributed during the day (from 00:00 to 22:00, every 2 hours). Figure 6 shows the average precision of the MaxTimeSlot (see Figure 6A) and the MaxSumSlot strategy (see Figure 6B) at 12 time instants of the day. Figure 6A confirms the fact that the MaxTimeSlot strategy performs better than the MaxSumSlot strategy. In general, both the MaxTimeSlot and the MaxSumSlot strategy expose an average precision that is rather low for the predictions made during the early morning (i.e., from 02:00 to 08:00), i.e., about $0.75 \pm 0.04$ for MaxTimeSlot and $0.68 \pm 0.03$ for MaxSumSlot. Instead, the predictions concerning the afternoon or the evening (i.e., from 10:00 to 22:00) have higher average precision (i.e., about $0.91 \pm 0.02$ for MaxTimeSlot and $0.84 \pm 0.03$ for MaxSumSlot). The analysis we con-
FIGURE 6  Analysis of the average precision at different instants of time of a day. The 95% coefficient interval is shown as a vertical line segment on each box. A, MaxTimeSlot user selection strategy; B, MaxSumSlot user selection strategy

ducted at macroscopic and microscopic levels for both the MaxTimeSlot and MaxSumSlot strategies reveals a very complex relationship between the history size used by the linear predictor and the specific time of the day for which we want to predict the user availability. In our simulations, we observe that for predictions occurred between 02:00 and 06:00, a longer history size is more beneficial, whereas the predictions occurred between 08:00 and 12:00 benefit from a shorter history size. Indeed, users who are online during the night (from 02:00 to 06:00) exhibit regular availability patterns that are repeated over time, hence benefiting from a 14 days of history. In addition, we observe that this behavior is not dependent on the selection strategy used by the predictor because the same trend occurs for both MaxTimeSlot and MaxSumSlot strategies.

Figures 6A and 6B show that from 08:00 to 12:00 the longer history size negatively affects the performance in predicting user availability. This means that the user availability information older than 7 days does not reflect the current behavior of the user anymore. Finally, Figure 6A shows that predictions occurred between 14:00 and 00:00 expose similar average precision, regardless of the size of the availability history used by the linear predictor. The behavior of the users during this period of the day is very heterogeneous, and it is characterized by a mixed pattern where longer and shorter history sizes may have similar benefits. In particular, this result suggests that users may have very different features from each other, and the availability predictions in this time interval should use the history size that best matches the needs of the predicted user.

7.4.3 Size of the Prediction Window

As a further step, we characterize how the performance of the predictor is related to the size of the future time interval to predict. For this purpose, we evaluated the precision of the linear predictor over future time intervals of different sizes (y): 30 minutes, 1 hour, and 2 hours. Figure 5B shows the average precision of the Random, MaxTimeSlot, and MaxSumSlot strategies with respect to the different sizes of the predicted time interval. The graph confirms the global trend, showing that the MaxTimeSlot strategy is more successful and effective than the MaxSumSlot and Random strategies. In particular, the prediction of a period of time in the future that exceeds 30 minutes leads to an increase in the average precision of the MaxTimeSlot and MaxSumSlot strategies. In fact, we observed that the number of distinct users selected by the two strategies increases with the size of the future time interval to predict as more users are needed to cover a longer period of time. We focused our attention on the analysis of the precision at different time instants of the day. Figure 7 shows the average precision of the MaxTimeSlot (Figure 7A) and MaxSumSlot strategy

FIGURE 7 Analysis of the average precision at different instants of time of a day and for different numbers of predicted time instants. The 95% coefficient interval is shown as a vertical line segment on each box. A, MaxTimeSlot user selection strategy; B, MaxSumSlot user selection strategy
(Figure 7B) at different time instants of the day and for different sizes of the time interval to predict. We can notice that the majority of the time instants exhibit an increasing trend: the highest average precision is achieved in the case of predictions for the next 2 hours while the lowest precision value is reached by predictions made on a prediction window of size equal to 30 minutes. It is interesting to notice that, for both the strategies, the predictor exhibits a different behavior at time instant between 10:00 and 12:00 of the day, where the highest precision value is achieved by predicting a future time period of 1 hour. We explain this because most users start to connect regularly to the OSN in the afternoon, whereas during the late morning, i.e., starting from 10:00, the users perform sporadic and short connections probably preceding the lunchtime.

7.4.4 User transitions

Another very important aspect for the performance of the prediction is the number of distinct users selected by each strategy and how often these users are going to disconnect from the OSN during the predicted period of time. Figure 8A shows the average number of distinct users selected by the MaxTimeSlot and MaxSumSlot strategies. The MaxTimeSlot strategy selects more users than the other strategies because the prediction of a time slot is independent from the previous or the next ones. As for instance, in the worst case, the MaxTimeSlot strategy could select at least one distinct online user in each time slot of the predicted time interval. Moreover, the average number of distinct users selected by the MaxSumSlot strategy is about half of those of the MaxTimeSlot strategy. In addition, when the status of a user changes from online to offline, it is necessary to perform some actions that ensure a seamless handover of the service. For this purpose, we computed the frequency with which the status of a predicted user changes from online to offline during the predicted period of time (i.e., 30 minutes, 1 hour, or 2 hours) occurred for the MaxTimeSlot and MaxSumSlot strategies (with 14 days of history) since it has proven to be better than other. Figure 8B shows the average number of transitions occurred when predicting users’ availability both during the morning (AM) and the evening (PM). The graph clearly indicates that the average number of transitions occurred at AM is greater than the ones occurred at PM. This fact confirms that users’ transitions (from online to offline) occurs

---

**FIGURE 8** Analysis of the number distinct users and the number of transitions online-offline performed at different periods of the day and for different numbers of predicted time instants. The 95% coefficient interval is shown as a vertical line segment on each box. A, Average number of distinct users; B, Average number of online-offline transitions

**FIGURE 9** Frequency distribution of the transitions number of the MaxTimeSlot strategy occurred in the morning (Figure 9A) and in the evening (Figure 9B) for different numbers of predicted time instants. A, Number of transition at morning (AM); B, Number of transition at the evening (PM)
mainly when predicting users' availability at morning. Indeed, users connect to the OSN principally during specific time intervals of the day (e.g., after lunchtime). In contrast, availability patterns of users who appear to be online during the night (from 00:00 to 11:00) are not cyclic and occur occasionally over time. In particular, the number of transitions occurred for the MaxTimeSlot strategy is quite higher than those occurred for the MaxSumSlot strategy.

Figure 9 shows the frequency distribution of the number of transitions for different numbers of future time instants predicted. We can notice that the distribution of values is right (positively) skewed. In general, we observe that the number of online-offline transitions both at AM and PM increases as long as we try to predict extended periods of time in the future.

8.1 Concluding remarks

By using temporal information of a real OSN, we have found that availability patterns of a single individual have a strong impact on the prediction of user's availability. Furthermore, the study that we conducted uncovered a number of interesting findings related to the user behavior in OSNs. We showed that users connect to the OSNs frequently (about 90% of users perform at most 10 sessions per day) and for short periods of time (half of the sessions are shorter than 10 minutes). We define a set of predictors based on intervals, which can be useful to support the availability prediction problem. In particular, the linear predictor exploits the availability patterns of user $u$, during the past $x$ days, in order to predict the availability status of $u$ in future time intervals of size $y$. In addition, we have proposed and evaluated different selection strategies that allow to select the users that will be most probably online. The evaluations, based on data obtained from a Facebook application, have shown that the proposed approach ensures an average prediction rate of about 80% for the prediction of a future period of time of size less than 2 hours and by using past online status information related to the last 14 days. In general, availability patterns of users are very heterogeneous and depend on the time of the day to predict, the prediction window, and the history size. Figure 10 summarizes the configuration parameters that maximize the accuracy of the linear predictor at different times of the day. We can notice that, during the early morning (from 2 to 6), the best prediction accuracy is achieved by using long history size (14 days) and long prediction window (2 hours), whereas in the first middle of the day (from 8 to 12), a shorter history size (7 days) and prediction window (1 hour) result to have better performance. Instead, for the rest of the day (i.e., from 14 to 25), the proposed linear predictor exposes the best accuracy when a prediction window of 2 hours is used, regardless of the history size (7 or 14 days). Finally, a short prediction window (i.e., 30 minutes) is never used because it has the worst performance.

8.2 Limitations and future works

The linear predictor we defined has no adaptation strategy and it fails in predicting the users that will be online in case of unexpected events (such as weekends or holidays). However, dynamic adaptation strategies, such as those defined in the work of Dell’Amico et al., can be exploited to select the most appropriate coefficient vectors that minimize the number of incorrect predictions. As future work, we would like to investigate adaptive techniques that allow the linear predictor to react quickly to the occurrence of these specific patterns or unexpected events. We plan to introduce learning strategies that automatically adjust the weights given to the previous $x$ days so as to maximize accuracy of the linear predictor.

In addition, the proposed linear predictor was designed to consider the requirements of the DOSN scenario, where users connect for a short period of the time. However, we plan to extend the proposed predictor to consider a linear combination of several predictions capturing annual, monthly, weekly, or seasonal patterns. Furthermore, we would like to investigate the usage of our linear predictor in content distribution task. Indeed, selecting the users who will be available at a specific time interval in the future will let DOSNs to efficiently manage the distribution of content and data to the different replicas.
ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their valuable remarks that helped improve the quality of this manuscript.

ORCID

Andrea De Salve http://orcid.org/0000-0003-1691-7182

REFERENCES


https://diasporafoundation.org/


How to cite this article: De Salve A, Guidi B, Mori P. Predicting the availability of users’ devices in decentralized online social networks. Concurrency Computat: Pract Exper. 2017:e4390. https://doi.org/10.1002/cpe.4390