# Modeling User Interactions in Social Communication Networks with Variable Social Vector Clocks

Ta Yuan Hsu
*University of Illinois at Chicago*
*Chicago, IL, 60607-7053, USA*
*Email: thsu4@uic.edu*

Ajay D. Kshemkalyani
*University of Illinois at Chicago*
*Chicago, IL, 60607-7053, USA*
*Email: ajay@uic.edu*

Min Shen
*University of Illinois at Chicago*
*Chicago, IL, 60607-7053, USA*
*Email: mshen6@uic.edu*

***Abstract*—Social communication networks have been widely investigated in recent years. From fine-grained temporal information's point of view, *social vector clock* (SVC) is a useful mechanism to track the most recent communication with all other local peers in a social network. A modification of conventional social vector clocks has been proposed to deal with the issue of poor scalability without keeping whole temporal views [1]. In this paper, we propose an idea of variable social vector clocks (VSVCs) and the corresponding incremental algorithm. Not only does it maintain the lower bound of how out-of-date each peer can be with respect to others, but it also considers the shortest friendship separation to restrict how far information may be transmitted along time respecting paths. Unless this separation is bounded to be infinite, some communication messages could be lost implicitly. We also focus on studying message inconsistency and reachable in-degree communication distribution in several social networks based on variable social vector clocks.**

***Keywords*-variable social vector clocks; message loss rate; reachable in-degree distribution**

## I. Introduction

The analysis of online communication for large social networks and social supports has attracted wide attention in a variety of areas over the past few decades. For example, larger social networks act as tubes for information flows and messaging communication [2][3]. Online social networks, such as Facebook, provide a common platform to connect individuals in terms of some characteristic attributes, such as interests, activities, ideas, backgrounds, or events, and allow them to interact with each other over the Internet. A large amount of evidence shows that social network services have already influenced the habit of using Internet and become a major part of our daily lives all over the world [4][5]. It also insinuates that the degree of social networking communications grows quickly over time. Actually, it has been a challenge to extract complete data flows for event-driven communication in a large-scale network over long periods of time. Since event-driven data flows may be highly relevant with the spacing and ordering of events, the fine-grained temporal approach has been proposed to figure out the dynamic interactions in a social network through a temporal framework [6].

Applying the fine-grained temporal view, some researchers have begun to focus on information path latency and indirect message exchanging in social communication networks with the utilization of vector clocks from the field of distributed systems. Vector clocks were conceptually introduced by Mattern [7]. A vector clock in a concurrent system can be used to track the lower bound of how recent each process's state is associated with any other one at a given time. With a partial ordering of events in such a system, vector clocks can detect whether a given pair of events are causally related.

Kossinets et al. [8] applied the concept of vector clocks to social networks, where they introduced a framework of Social Vector Clocks (SVCs). Through SVCs, communication flows were visualized over time [9]. It has also been used to formulate a temporal framework to figure out the structure of information pathways [8]. Ignoring the subject matter of information or messages, an ordering of time-stamps can provide a global view of user interactions. The SVC is an important mechanism capable of maintaining the contents and timestamps of the latest communications for each peer. As with Lamport timestamps, communication messages contain the local state of the sending process's logical clock. Consider an example as shown in Figure 1. Suppose that seven members are discussing some issues about traveling through a series of direct or indirect communication messages, such as user-targeted mention tweets, over four days. Let peer A be a coordinator responsible for collecting the other six members' opinions to make a conclusion. Peer A initially sent the first message to peer B at 11PM on Tuesday. Subsequently, peer B could send his idea and peer A's opinion to peer C. Finally, peer A would receive all indirect information from peers B, C, D, E, and F and the message from peer G at 6PM on Friday. Although peer A did not have direct communication with any peers other than peer G and B, peer A is still aware of the whole knowledge associated with all members' thoughts on Friday evening by indirect message exchanges using the properties of SVCs. The reason is that all indirect up-to-date communications between any two peers can be retained by SVCs, but with quadratic space requirements. However, a

Authorized licensed use limited to: University of Illinois at Chicago Library. Downloaded on June 22,2020 at 22:44:07 UTC from IEEE Xplore. Restrictions apply.

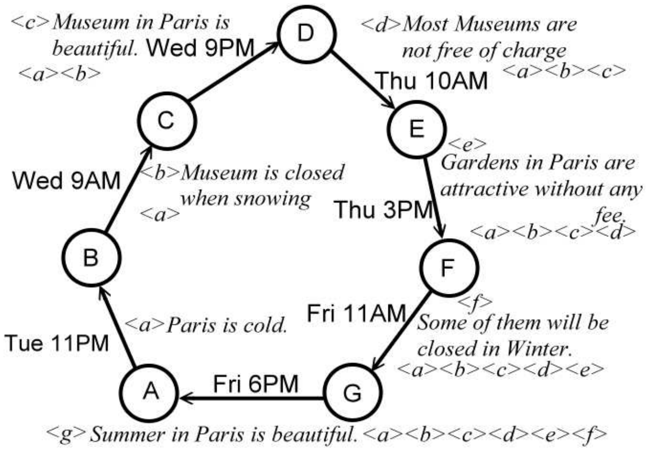
<c> Museum in Paris is beautiful. Wed 9PM
<a><b>
D
<d> Most Museums are not free of charge
Thu 10AM
<a><b><c>
C
E
<e> Gardens in Paris are attractive without any fee.
<a><b><c><d>
Wed 9AM
<b> Museum is closed when snowing
<a>
Thu 3PM
B
F
<f> Some of them will be closed in Winter.
<a><b><c><d><e>
Tue 11PM
<a> Paris is cold.
Fri 11AM
A
Fri 6PM
G
<g> Summer in Paris is beautiful. <a><b><c><d><e><f>

Figure 1. Illustrative example of social network communication for seven people. $< \cdot >$ denotes a communication message or a piggybacked message

large proportion of indirectly exchanged messages received may be meaningless for most peers in a social network. For example, if peer A informed another peer H (suppose that peer H is peer A's father) something independent of traveling issue at 7AM Saturday, peer H would still receive all indirect communication messages shown in Figure 1, even though peer H was unconcerned with those messages. As illustrated in this example, it is of no practical use to receive all peers' indirect updated messages for analyzing user interactions using the conventional SVCs in social groups. In [10], it is shown that cognitive constraints may not impact the size of the social network. One modification to the SVCs, along with shortest time-respecting paths, has been proposed to formulate the fine-grained temporal features applicable to large-scale social interaction networks with better scalability for link prediction [1]. The number of reachable indirect incoming messages is delimited by the minimum number of hops ($\mu$) from the source to the receiver on time-respecting paths; some messages may be lost and inconsistent when $\mu <$ the number of peers. As shown in Figure 1, if $\mu$ is bounded to be 1 such that any indirect updates will be discarded, peer A may obtain inconsistent information. This observation implies that the message loss rate must be affected by the value of $\mu$.

In this paper, we extend the modification of SVCs to be variable social vector clocks (VSVCs) and apply VSVCs to quantitatively analyze the influence of the restriction of $\mu$ on the reachable information. Here, we focus on two small social groups from Twitter, where participants can explicitly dispatch messages to targeted receivers. The two groups are a list of UK athletes organized by *The Telegraph* and a list of past and present *MLB* players. We also present the reasonable compromise of message loss rate and memory space requirement, and then the degree of separation for incoming reachable messages.

## II. VECTOR CLOCKS

In this section, we first briefly introduce the concept of conventional vector clocks (VC) in distributed systems [7]. We then give a simple overview about the framework of SVCs and the modification of SVCs. Last, we explain how to exploit the framework of variable social vector clocks (VSVC) in social networks.

### A. Conventional Vector Clocks

Basically, to establish vector clocks in the system, each process $P_i \in P$ (the set of processes in the system) maintains a vector clock $V_i$ of $N$ (the number of processes in the system) integers, which is updated by the following rules.

1) Before an internal event happens at process $P_i$, $V_i[i] = V_i[i] + 1$.
2) Before process $P_i$ sends a message, it first executes $V_i[i] = V_i[i] + 1$, then it sends the message piggy-backed with $V_i$.
3) When a process $P_j$ receives a message with timestamp $U$ from $P_i$, it executes
$\forall k \in [1 \dots N], V_j[k] = \max(V_j[k], U[k])$;
$V_j[j] = V_j[j] + 1$; before delivering the message.

Conventional vector clock in a distributed system is used to track the causal relation ($\prec$) between two events that happened in the system by comparing their corresponding vector clock timestamps, i.e., $e_i \prec e_j \Leftrightarrow V_{e_i} < V_{e_j}$, where $V_{e_i} < V_{e_j}$ means $\forall a \in [1, N], V_{e_i}[a] \leq V_{e_j}[a]$ and $\exists b \in [1, N]$ such that $V_{e_i}[b] < V_{e_j}[b]$. Furthermore, the conventional vector clock can also be expressed as a multi-variate function on a 3-tuple $(t, s, r)$, which is (time, sender, receiver). $P_i$'s temporal view of every process in the system at time $t$ is defined as a function $\phi_{i,t} = (\phi_{i,t}(j) : j \in P)$ where $\phi_{i,t}(j)$ is $P_i$'s temporal view of a particular process $P_j$ at time $t$. Here, $\phi_{i,t}(j)$ corresponds to the $j$th entry of $P_i$'s local vector timestamp $V_i$ when $V_i[i] = t$.

### B. Social Vector Clocks

The SVC updating approach basically obeys the mechanism of the conventional vector clocks without any communication delay and with only one global time clock. When receiving an incoming event ($E_t$) sent on time slice $t$ from $P_i$, the timestamp of the receiver $P_j$'s temporal view of the $i$th entry has to be equal to that of the $j$th entry of $P_j$'s vector clock. The left of Figure 2 illustrates how to update the entries on SVCs. For example, when $P_3$ receives $E_{t_4}$ sent from $P_2$ at $t_4$, $V_3[2] = t_4$ and $V_3[3] = t_4$ as well. By the rule 3 in the conventional vector clocks, $V_3[1]$ and $V_3[4]$ will be $t_3$ and $t_2$, respectively.

In a $N$-peer social group, SVCs require O($N^2$) space globally to maintain the latest temporal views. Under the piggybacking approach, each peer will soon get a large number of indirect updating messages from other peers. However, most of receivers do not have any direct communication and their social-connection steps are too far away

Authorized licensed use limited to: University of Illinois at Chicago Library. Downloaded on June 22,2020 at 22:44:07 UTC from IEEE Xplore. Restrictions apply.

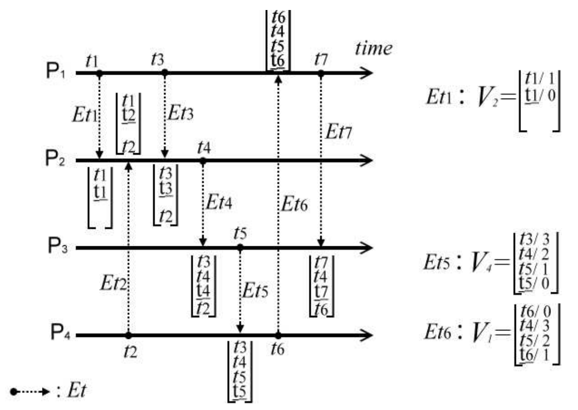
Figure 2. Illustrative example of social network communication for four peers with the SVCs.

from each other. In other words, it may not be efficient to directly utilize SVCs, because too much impractical information will be generated.

A modification of SVC updating approach has been proposed in [1], where a parameter $\mu$ is introduced as the minimum number of hops between a pair of peers along time-respecting paths. The illustration of different values assigned to $\mu$ is as follows:

$\mu = \infty$ ( it is sufficient for $\mu$ to be $N$-1 in a $N$-peer social group): it is equivalent to the conventional SVC updating approach, considering unlimited indirect communication spread without self-looping updating, as shown in Figure 2.

$\mu = 1$ : it only focuses on direct friendship communication. A receiver can update based on the incoming message if and only if the corresponding sender ever directly interacts with the receiver.

$\mu = 2$ : not only does it allow all direct updating with $\mu = 1$, but it also considers friend-of-friend communication. That is, a receiver $j$ can accept an incoming piggybacked message, indirectly come from peer $k$ and directly sent from peer $i$, if and only if peer $k$ has ever directly interacted with the receiver $j$.

## C. Variable Social Vector Clocks

In order to consider all different reachable distances of friendship, we define a universal framework of the variable social vector clocks in social networks.

Not only does a process keep the latest temporal views for each peer with respect to others, but it also computes the shortest friendship distances for each pair of peers: the $j$th entry of peer $i$'s social vector clock ($V_i[j]$) maintain two records. The first record ($V_i[j].time$) still tracks the temporal view of peer $i$ on peer $j$ and the second record ($V_i[j].dist$) is the shortest time-respecting path. When an incoming event $E_t$ occurs sent from peer $j$ to peer $i$ at time slice $t$, then

$V_i[j].time = E_t.timestamp$

$V_i[k].dist = \min(V_j[k].dist + 1, V_i[k].dist)$ when $k \neq j$

The right of Figure 2 illustrates some examples of applying VSVCs to the case in the left of Figure 2. Let us consider $V_2$ as the VSVC of $P_2$. When receiving $E_{t1}$, the first entry of $V_2$ contains $t_1$ (the latest sending timestamp of $P_1$) and 1 (the value of $\mu$) from $P_1$ to $P_2$. In this paper, we use the framework of VSVCs to analyze how the social communication is affected by $\mu$.

## III. Methodology

In order to investigate the impact of the hop-constrained paths on social communications, we design an efficient accumulative vector clock updating algorithm keeping track of the mimimum number of hops the information travels between a pair of sender and receiver. Whenever one communication event $E_t$ happens, triggered from a sender peer ($E_t.sender$) to a receiver peer ($E_t.receiver$), the receiver peer's vector clock needs to be updated. For each update, the shortest distances from all peers to the receiver peer need to be computed as well. The algorithm pseudo-code of $VariableSocialVectorClocks$ is shown in Algorithm 1. A sequence of $E_t$ ($0 \leq t \leq T$) is treated as the input. In lines 1-4, the timestamp vector ($V_i.time$) and distance vector ($V_i.dist$) for each peer $i$ with respect to any other peer are initialized. Each peer maintains its own vector clock with the latest information with respect to all other peers. The value of $DIST_i[k]$ gives how many messages are indirectly/directly delivered to peer $i$ in the minimum number of hops of $k$ when $k \geq 1$, where $k$ is the shortest distance between a pair of peers. $V_x[y].time$ represents the latest temporal view of peer $x$ on peer $y$; $V_x[y].dist$ maintains the shortest distance from $y$ to $x$.

There are four different cases for updating. Lines 7-17 deal with the first case when both sender's and receiver's vector clocks have been active. 'Active' implies that a peer has received at least one direct incoming information from any other peer. This case occurs most often, such as $E_{t7}$ shown in Figure 2. $P_1$ becomes 'active' after time $t_6$ and $P_3$ being 'active' after time $t_4$. Lines 18-23 correspond to the second case, such as $E_{t2}$, where $P_4$ has not been active until $t_5$ but $P_2$ has become active since $t_1$. Lines 24-28 treat the third case like $E_{t6}$ contrary to the second case. Lines 29-32 illustrate the fourth case, such as $E_{t1}$, where neither of $P_1$ and $P_2$ has been active at $t_0$.

Note that when a receiver $j$ finishes updating timestamps and distances, $DIST_j[]$ also needs to be updated by lines 17, 23, 28, 32 for different cases. Let's consider the example of $E_{t2}$ as shown in Figure 2. After receiving $E_{t1}$, $V_2$ should be $[t1/1, t1/0, \perp, \perp]$. $DIST_2[0,1,2,3]$=$[1,1,0,0]$. When receiving $E_{t2}$, $V_2$ becomes $[t1/1, t2/0, \perp, t2/1]$. Based on lines 27-28, when $k$=2, $V_2[2].dist$ is '0'. $DIST_2[0]$++ has to run one time and will be equal to '2'. When $k$=0 and 3, $V_2[0].dist$ and $V_2[3].dist$ are both '1'. $DIST_2[1]$++ will run two times and then becomes '3'. Lines 32-34 accumulate $DIST_i[]$ from each peer into a global $DIST[]$.

Authorized licensed use limited to: University of Illinois at Chicago Library. Downloaded on June 22,2020 at 22:44:07 UTC from IEEE Xplore. Restrictions apply.

**Algorithm 1:** VariableSocialVectorClocks

```
Input: $E_0$,..., $E_T$
Output: $V_0$,...,$V_{N-1}$ and $DIST[]$
for $i \leftarrow 0$ to $N-1$ do
    $V_i.time \leftarrow [\perp,...,\perp]$;
    $V_i.dist \leftarrow [\perp,...,\perp]$;
    $DIST_i \leftarrow [0,...,0]$;
while $t \leq T$ do
    $j \leftarrow E_t.receiver$; $i \leftarrow E_t.sender$;
    if $V_i$ and $V_j$ have been active then
        $V_j[j].time \leftarrow E_t.time$;
        for $k \leftarrow 0$ to $N-1$ but $k \neq j$ do
            if $V_i[k] \neq \perp$ and $V_j[k] \neq \perp$ then
                $V_j[k].time \leftarrow$ max($V_i[k].time$,$V_j[k].time$);
                if $V_i[k].dist < V_j[k].dist$ then
                    $V_j[k].dist \leftarrow V_i[k].dist$+1;
            else if $V_i[k] \neq \perp$ and $V_j[k]$ is $\perp$ then
                $V_j[k].time \leftarrow V_i[k].time$;
                $V_j[k].dist \leftarrow V_i[k].dist$+1;
            $DIST_j[V_j[k].dist]$++;
    else if $V_i$ is active but $V_j$ has not been active then
        $V_j[j].time \leftarrow E_t.time$; $V_j[j].dist \leftarrow 0$;
        for $k \leftarrow 0$ to $N-1$ but $k \neq j$ do
            $V_j[k].time \leftarrow V_i[k].time$;
            $V_j[k].dist \leftarrow V_i[k].dist$+1;
            $DIST_j[V_j[k].dist]$++;
    else if $V_j$ is active but $V_i$ has not been active then
        $V_j[i].time \leftarrow E_t.time$; $V_j[j].time \leftarrow E_t.time$;
        $V_j[i].dist \leftarrow 1$;
        for $k \leftarrow 0$ to $N-1$ but $k \neq j$ do
            $DIST_j[V_j[k].dist]$++;
    else if both $V_i$ and $V_j$ have not been active then
        $V_j[i].time \leftarrow E_t.time$; $V_j[j].time \leftarrow E_t.time$;
        $V_j[i].dist \leftarrow 1$; $V_j[j].dist \leftarrow 0$;
        $DIST_j[1] \leftarrow 1$;
    $t$++;
for $k \leftarrow 1$ to $N-1$ do
    for $i \leftarrow 0$ to $N-1$ do
        $DIST[k] \leftarrow DIST[k]$+$DIST_i[k]$;
```

## A. Message Loss Rate

$VariableSocialVectorClocks$ is an incremental algorithm to update the latest timestamp and the shortest distance at each time slice $t$. In an $N$-peer social network group, when $\mu$ is bounded to be $N$-1, it is equivalent to the conventional social vector clock algorithm with unlimited communication spread. Obviously, the total number of direct or indirect updating messages without losing any communication is:

$$DIST[k] = \sum_{i=0}^{N-1} DIST_i[k]; (N_M = \sum_{k=1}^{N-1} DIST[k]) \quad (1)$$

When the upper bound of $\mu < N$, based on $DIST[1 \sim \mu]$, we can compute the message loss rate corresponding to the specified value of $\mu$:

$$R_L(\mu) = 1 - \frac{\sum_{k=1}^{\mu} DIST[k]}{N_M} \quad (2)$$

$R_L(\mu)$ can be viewed as the space saving rate. Interestingly, it decreases as the upper bound of $\mu$ increases. With this view, we can gain insight into how to determine the upper bound of $\mu$ to efficiently utilize memory space without losing reasonable communication in a social network.

## B. Reachable in-degree distribution

In our methodology, we also study the reachable in-degree distribution using the output vector clocks $\{V_0,..,V_{N-1}\}$ at the latest time slice $T$ in Algorithm 1. We want to learn about how many peers ($N_p$) whose sending information can reach one peer $p$ in the steady state. Obviously, $N_p$ is subject to $\mu$ based on the up-to-date value of $V_p[i].dist$. By summing up the value of $x_i$ ($x_i$: a unit step function)

$$N_p(\mu) = \sum_{i=1}^{N} x_i \begin{cases} x_i = 1 & \text{if} \quad V_p[i].dist \leq \mu; p \neq i \\ x_i = 0 & otherwise \end{cases} \quad (3)$$

If $x_i$=1, a message sent from peer $i$ can reach peer $p$; otherwise, it is unreachable even though there exists a connection path. Due to the variation of individual peers, we normalize the value of $N_p$ for each peer:

$$N_{ave}(\mu) = \frac{\sum_{p=0}^{N-1} N_p(\mu)}{N} \quad (4)$$

We refer to $N_{ave}(\mu)$ as the 'average reachable in-degree distribution number'. Apparently, the value of $N_{ave}$ positively depends on the bound of $\mu$. Therefore, the average reachable in-degree rate can be defined as:

$$R_{ave}(\mu) = \frac{N_{ave}(\mu)}{N} \quad (5)$$

If social communication is in steady-state, $R_{ave}(\mu)$ should be close to 1-$R_L$. Furthermore, we also use $N_{ave}(\mu)$ to evaluate the degrees of separation for individual social networks in our experiments based on *the six degrees of separation* proposed by Frigyes Karinthy in 1929.

## IV. Results and Evaluation

In this section, we present a brief overview of our system architecture for experiment setup, dataset configuration, and

Authorized licensed use limited to: University of Illinois at Chicago Library. Downloaded on June 22,2020 at 22:44:07 UTC from IEEE Xplore. Restrictions apply.

our experimental results. The brief overview is followed by the evaluation and discussion.

### A. Experiment Setup

The system infrastructure is built along with the entities outside the system that it interacts with and a description of the interfaces between these entities. The datasets we consider come from Twitter. We extract tweets to collect communication data into our dataset using a REST functional implementation through the twitter4j package based on certain filters. To support our methodology, $VariableSocialVectorClocks$ is applied to two datasets to illustrate the influence of the separation degree constraint on social communication.

### B. Datasets

In the Twitter data that we analyze, we filter them to include only two targeted forms of communication. The first one occurs in the form of retweets (where one user rebroadcasts another user tweets). The second one happens in the form of user mentions (where the @ symbol is used to explicitly refer to a specific user). If there exist more than one user in a tweet, we turn it into as many events as there are multiple users represented in the tweet.

**Twitter UK Olympics Data**: The Olympics dataset covers Twitter communication among a set of 476 UK Olympic athletes over the course of the 4 years until now, including about 940,000 tweets. It is based on a list of UK athletes organized by $The\ Telegraph$.

**Twitter MLB Data**: The MLB dataset includes Twitter communication among a list of 550 past and present MLB players in 2013 on Twitter, containing about 660,000 tweets. It is based on a list of Major League Baseball players organized by $MLB$.

### C. Experiment Evaluation and Results

For the UK dataset, the total number of messages that can be captured without any separation constraints is 48,797,781. For the MLB dataset, the total number of messages delivered is 25,154,068. Figure 3 shows the relationship between the minimum number of hops($\mu$) and the message loss rate $R_L$. The detailed results are shown in Table I. If $\mu$ reaches more than eleven, the message loss rates will be close to or less than 1%, which are ignored in Table I. When the upper bound of $\mu$ is limited to 'two', the memory space requirements, proportional to 1-$R_L$ , can be reduced up to about 0.69 $\sim$ 0.81. In several social networks, the case of $\mu$=2 has been considered and shown to be significant to the impact of information brokerage activities [11]. Based on the observation of $R_L$ in Table I, it shows that the variable social vector clock updates can effectively improve the memory space utilization and can more closely approximate how far information should be tracked in real social networks. When $\mu$ is bounded to be six, the values of $R_L$ (space saving rate) are about 6.5% and 8.4%, respectively. In other words, more than 90% of direct or indirect communication messages may be maintained. It implies that when $\mu$ = 6, the modification of social vector clocks almost act as the conventional ones.

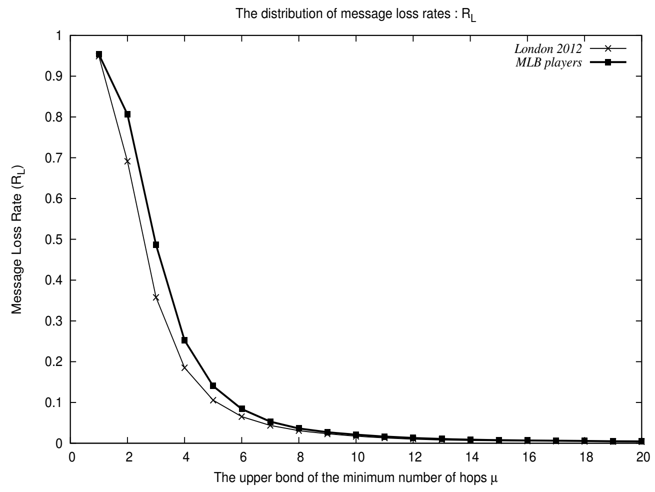

Figure 3. Message Loss Rate

Table I
MESSAGE LOSS RATES $R_L(\mu)$ ; $\mu$ IS THE MINIMUM NUMBER OF HOPS; $N(\mu)$ IS THE NUMBER OF RECEIVED MESSAGES

| $\mu$ | $London$ 2012 | | $MLB\ players$ | |
|---|---|---|---|---|
| | $R_L(\mu)$ | $N(\mu)$ | $R_L(\mu)$ | $N(\mu)$ |
| 1 | 0.9491 | 2,482,898 | 0.9534 | 694,949 |
| 2 | 0.6915 | 15,055,874 | 0.8062 | 4,471,680 |
| 3 | 0.3576 | 31,345,425 | 0.4864 | 12,675,947 |
| 4 | 0.1851 | 39,766,327 | 0.2522 | 18,684,643 |
| 5 | 0.1055 | 43,649,830 | 0.1403 | 21,554,702 |
| 6 | 0.0653 | 45,611,294 | 0.0840 | 22,999,239 |
| 7 | 0.0436 | 46,670,688 | 0.0531 | 23,792,599 |
| 8 | 0.0308 | 47,292,706 | 0.0363 | 24,222,348 |
| 9 | 0.0227 | 47,688,174 | 0.0270 | 24,461,355 |
| 10 | 0.0167 | 47,980,881 | 0.0210 | 24,615,896 |
| 11 | 0.0130 | 48,161,113 | 0.0163 | 24,736,486 |

For the analysis of reachable in-degree distributions ($N_{ave}(\mu)$), as per the definition in Section 3, Figure 4 shows the distributions of $N_{ave}(\mu)$ for these two groups. In [1], the authors only considered direct friendship ($\mu$=1) and friendship-of-friendship ($\mu$=2) in their vector clock framework. Several social networks, such as Facebook, Twitter, and Google+, also allow direct friends and friends-of-friends to communicate with each other. Based on the results shown in Table II, when $\mu$ is bounded to be two, each peer can communicate with about 30% $\sim$ 20% of peers on average in these two social groups under steady-state situations. The results are both consistent with the observations of the message loss rate. In terms of the theory of six degrees of separation, anyone or any individual in the real world can be connected to each other on the planet through a chain of acquaintances with six or fewer intermediaries. As a result, when the minimum number of hops ($\mu$) is bounded

Authorized licensed use limited to: University of Illinois at Chicago Library. Downloaded on June 22,2020 at 22:44:07 UTC from IEEE Xplore. Restrictions apply.

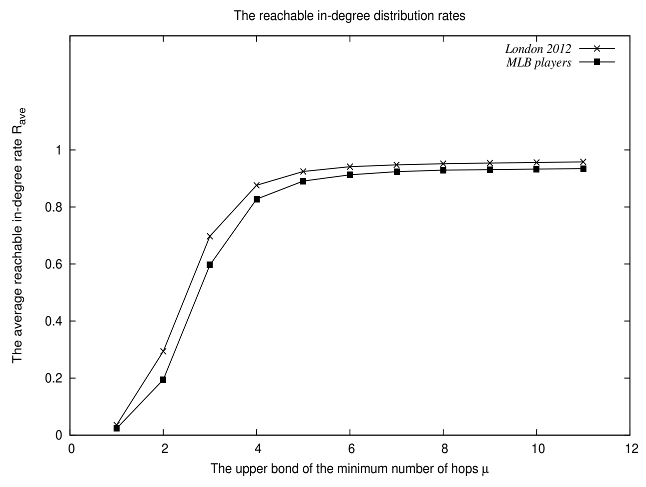

Figure 4. The Reachable In-degree Distributions

Table II
THE AVERAGE REACHABLE IN-DEGREE DISTRIBUTION NUMBER $(N_{ave}(\mu))$ AND RATE $(R_{ave}(\mu))$

| $\mu$ | $London$ 2012 | | $MLB\ players$ | |
|---|---|---|---|---|
| | $N_{ave}(\mu)$ | $R_{ave}(\mu)$ | $N_{ave}(\mu)$ | $R_{ave}(\mu)$ |
| 1 | 17 | 0.035714 | 13 | 0.02363 |
| 2 | 140 | 0.294118 | 107 | 0.194545 |
| 3 | 332 | 0.697479 | 328 | 0.596364 |
| 4 | 417 | 0.87605 | 455 | 0.827273 |
| 5 | 440 | 0.92437 | 490 | 0.890909 |
| 6 | 448 | 0.941176 | 502 | 0.912727 |
| 7 | 451 | 0.947479 | 508 | 0.923636 |
| 8 | 453 | 0.951681 | 511 | 0.929091 |
| 9 | 454 | 0.953782 | 512 | 0.930909 |
| 10 | 455 | 0.955882 | 513 | 0.932727 |
| 11 | 456 | 0.957983 | 514 | 0.934545 |

to be six, everyone can communicate with all the people in a social networking community. Intuitively, 'six' is applicable to evaluate our experimental results in the reachable in-degree distributions. Interestingly, as shown in Table II when $\mu$ = 6, $R_{ave}$ is more than 90% of peers in both cases. It reasonably implies that each peer can almost communicate with most of the peers within six steps in the same social group.

## V. CONCLUSIONS

In this paper, we presented a framework of $VariableSocialVectorClocks$ to incrementally update VSVCs. We quantitatively analyzed the influence of the constraint of the shortest friendship separation on the message loss rates and reachable in-degree distributions using two Twitter social network groups. Based on the friendship policy in several social networks, where they allow only direct friends and friends-of-friends to communicate with each other, we focus on the minimum number of hops ($\mu$) of 1 and 2 in our experiment evaluation. From the results of $R_L$, we observe that when $\mu$ is bounded to be 'two', the memory space requirements can be effectively reduced up to 70% $\sim$ 80%. Taken together with this restriction, the vector clock updates can become more efficient for the memory space utilization and more closely approximate how far information would be tracked along time-respecting paths.

Our experiment results also show that when $\mu$ is bounded to be six, the average reachable in-degree distribution rates ($R_{ave}$) in both social networking groups are statistically more than 90% (i.e., it can almost exchange messages with all peers in a social group). Besides, when the upper bound of $\mu$ is 'six', the message loss rates ($R_L$) in both experimental settings are less than 10%. It means that most of messages in social vector clocks can be retained. It is consistent with the results of $R_{ave}$. Furthermore, when $\mu$ is 2, the message loss rates ($R_L$) are respectively 70% and 80% (Table I) in agreement with the corresponding average reachable in-degree distribution rates ($R_{ave}$).

In our future work, we plan to integrate into social communication aggregation analysis with variable social vector clocks and develop a reasonable user interaction model.

## REFERENCES

[1] C. Lee, B. Nick, U. Brandes, and P. Cunningham, "Link prediction with social vector clocks," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13, Apr. 2013, pp. 784–792.

[2] B. Huberman and L. Adamic, "Information dynamics in the networked world," in *Complex Networks*, E. Ben-Naim, H. Frauenfelder, and Z. Toroczkai, Eds. Springer, 2004, vol. 650, pp. 371–398.

[3] M. Granovetter, "The strength of weak ties," *Am. J. Sociol.*, vol. 78, no. 6, pp. 1360–1380, 1973.

[4] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09, 2009, pp. 807–816.

[5] B. K. Lewis and C. Nicholes, "Social media and strategic communication: A two-year study of attitudes and perceptions about social media among college students," *Public Relations Journal*, vol. 6, no. 4, 2012.

[6] P. Holme and J. Saramäki, "Temporal networks," *Physics Reports*, vol. 519, no. 3, pp. 97–125, 2012.

[7] F. Mattern, "Virtual time and global states of distributed systems," *Proceedings of the Parallel and Distributed Algorithms Conference*, pp. 215–226, 1988.

[8] G. Kossinets, J. Kleinberg, and D. Watts, "The structure of information pathways in a social communication network," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08, 2008, pp. 435–443.

[9] M. Harrigan, "Using vector clocks to visualize communication flow." in *ASONAM*, N. Memon and R. Alhajj, Eds. IEEE Computer Society, 2010, pp. 241–247.

[10] R. M. Dunbar and R. Hill, "Social network size in humans," *Human Nature*, vol. 14, no. 1, pp. 53–72, 2005.

[11] R. S. Burt, "Secondhand brokerage: Evidence on the importance of local structure for managers, bankers, and analysts," *Academy of Management*, vol. 50, no. 1, pp. 119–148, 2007.

Authorized licensed use limited to: University of Illinois at Chicago Library. Downloaded on June 22,2020 at 22:44:07 UTC from IEEE Xplore. Restrictions apply.