Link Pattern Prediction in Opportunistic Networks with Kernel Regression

Di Huang[†] Sanfeng Zhang[‡] Pan Hui[†] Zhou Chen[‡]
[†]The Hong Kong University of Science and Technology, Hong Kong
[‡]Key Laboratory of Computer Networks and Informatin Integration(Southeast University) Ministry of Education
dihuang@ust.hk

Abstract—**Opportunistic networks (OppNets) have emerged as prospective network architecture due to the popularization of mobile devices and little monetary cost. Routing is the major concern in designing OppNets. The main obstacle for routing protocol design in OppNets is little knowledge of future link patterns, which leads to blind and unpredictable packet forwarding behavior. To achieve better packet delivery rate, OppNets have to retain and deliver multiple copies of a message, which consumes more devices' energy and causes a lackluster OppNets service. To this end, we aim at the prediction of future link patterns to explore mobile connectivity. In this paper, we propose PreKR-the kernel regression based estimation framework for link pattern prediction. We initially extract best features that can represent the network evolution. Then we models historical structural features by kernel regression with the output of link probability. Experimental results show that our method outperforms state-of-the-art prediction methods up to 25%. We also find that both reachability prediction and high degree nodes prediction reach more than 90% accuracy. In the end, we propose heterogeneous architecture for PreKR deployment and investigate two prospective OppNets applications to show how PreKR improve system performance.**

I. Introduction

OppNets is indispensable when we perform tasks like mobile data offloading and mobile content sharing. Links often break down due to nodes mobility, so OppNets have to work in store-carry-and-forward mode to forward messages with copies [27]. This work mode has two main drawbacks: wasting time of mobile users and exhausting storage and energy of mobile devices. OppNets are unable to deny unreachable services due to no real-time responses to users, which definitely wastes much users' waiting time. Usually multiple copies of messages are stored and transmitted in OppNets to overcome blind forwarding at the cost of memory and power exhaustion in mobile devices. These drawbacks disable the popularization of applications like [28], [24], [14], [9].

Previous work mainly focus on forwarding efficiency by routing protocol design. They designed various routing protocols to pursue high performance routing. Some protocols devoted to promoting the delivery rate [11], [4], [17] while others tried to minimize delivery delay [10], [2]. They applied some nearly time-invariant metrics like average meeting time and link frequency as indicators of link generation probabilities. These stable metrics, however, bias the forwarding decision because they failed to capture intrinsic variations of links in such intermediately connected networks.

Studies in these years have directed to exploring transitory laws of OppNets. [6] exploited the transient contact patterns of opportunistic networks and [29] studied the impact of different altruism distributions in throughput. It formulated contact periods into predictable on-period and unpredictable 'off-period'. [25] discussed inherent laws in OppNets networks by breaking network snapshots into temporal communities and evaluating contributions of different types of nodes in content dissemination. They tried to predict future network link patterns but could not give exact quantifiable and intuitive results for data forwarding decision.

Now we directly devote to quantify transitory link probability in future time period. Future link probability of a node pair is not consistent but evolves over time according to current connection patterns. If we can get the exact estimation of link probability, we are able to make forwarding decisions dynamically. In this paper, we propose PreKR- the kernel regression based framework for link prediction in OppNets. Based on predicted link probability, we can establish future link patterns to discuss connectivity and reachability of the whole network.

In OppNets, we believe that future devices' mobility patterns evolve from current ones. Thus it is sound to assume that future links are determined by their current link patterns. Node pairs with similar network structure currently will share the same probability of propagating a link in the future. In PreKR framework, we exploit some social networks link prediction indices that represent the network structure as the baseline of our prediction work. Simple point estimation for these indices is not enough for such a mobile system, so we model the link probability via kernel regression. At last we rank the probabilities in descending order and select several highest ranked links as the predicted patterns. To deploy PreKR framework, we propose a three-layer architecture including OppNets layer, CellNets layer and Server layer.

The contributions in our paper are three-fold:

- To our best knowledge, it is the first time in OppNets to predict the future link probability dynamically. Predicted probabilities evolve with time and reflect the transitory connections which will greatly benefit forwarding decision.
- Based on the PreKR framework, we can give the predicted link pattern, which can evaluate the reachability of services. If packets are unreachable to destination,

the sender directly denies packet sending to avoid blind forwarding. Eventually energy and storage waste can be diminished.

- We propose the heterogeneous system architecture, which can not only realize PreKR but also send timely responses to content applicant. This will greatly improve user experience and save worthless waiting time by denial of unreachable service.

The reminder of the paper is organized as follows. We investigate related work in Section II. We introduce some of the state-of-the-art social network prediction in Section III and utilize them as the prediction baseline in PreKR framework. In Section IV we propose PreKR framework and try to lower the computation complexity. We present results in Section V. Finally we discuss how to deploy PreKR for prediction and how it can help application design in Section VI.

II. Related Work

Link prediction has been explored for long in both social networks and complex networks [15], [20]. They used the predicted links to make friend recommendation, email system monitoring and so on. [20] divide link prediction into two categories: similarity based prediction and model driven prediction. In similarity based approaches[15], indices like Common Neighbors, Adamic-Adar, Jaccard and Resource Allocation represent the closeness between node pairs and a high value of these indices means a high probability they will establish a link. In model driven approaches, however, link probability is calculated according to the network structure [8].

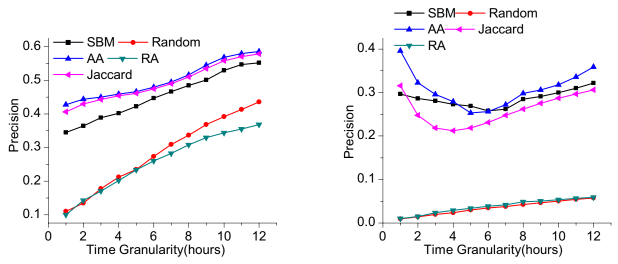
Recent years have also witnessed machine learning algorithms applied in prediction. For example Hasan *et al.* [1] compared several classical supervised learning algorithms and found that SVM with linear kernel perform best. Brouard *et al.* [3] focused on semi-supervised learning algorithm for link prediction. It could perform as well as the supervised learning algorithms by fetching little labeled data, which saved a lot of cost for link prediction.

Routing metrics in OppNets has been involved in prediction for many years [4], [16]. Maxprop [4] defined the probability with the contact times and select route with highest probability. Prophet [16] introduced the probability prediction mechanism to evaluate the forwarding utility. Node pair contact probability will be updated after their connections. Most of these metrics are naive and behave poor in link prediction. In other words, they are not good reflection of transitory link probabilities.

Existing work on both social networks and OppNets focuses on link prediction with static assumption or history data overlaying. In OppNets, we have to account for history fluctuation for transitory prediction of future links.

III. Prediction Baseline

Many social network prediction methods are based on social interaction structure. They believe that two nodes will link with each other if they are strongly social interactive. For example, if two people in a conference have many common friends, they are likely to share same research interests and connect with each other by their common friends. The number of common friends is the indicative score to judge the likelihood they will connect.



(a) SIGCOMM (b) UCSD

Fig. 1. Predicting Accuracy of Indices in OppNets Datasets.

In this section we choose four classical methods in social networks: Adamic-Adar (AA), Jaccard [15], Resource Allocation(RA) [30], and Stochastic Block Models(SBM) [8]. The first three methods calculate the score according to local graph structure while the last one calculate globally. These social scores can also partially represent OppNets link possibilities because mobile users are more likely to connect with a stronger social interaction. We apply them in two OppNets datasets (SIGCOMM and UCSD) to analyze their prediction performance and select the best index as one of the PreKR baseline.

It has been demonstrated in Figure. 1 that the link pattern can be inferred from AA score best. Adamic-Adar was first used to to infer the friend relations between two people by measuring the similarity of their homepages . They believe that the probability that they are friends is determined by common items on their homepage. Different items should contribute different to the probability. If a common item they share appears in low frequency in all homepages, they will be more likely to be friends because they both pay attention to this item and know with each other via this item. The local score between i and j is:

$$ls(i,j) = \sum_{z \in N(i) \cap N(i)} \frac{1}{\log k_z} \tag{1}$$

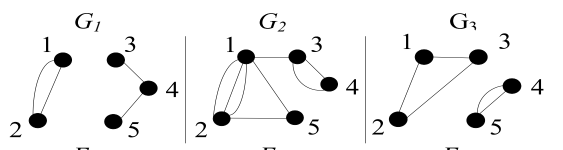
here $N(i)$ and $N(j)$ respectively denote the neighbors of i and j. k_z is the degree of their common neighbor z's degree.

Different from social networks, link patterns in opportunistic networks vary frequently due to the mobility of users. And each pair may contact several times in some time period. In some routing protocols, contact frequency (CF) is a fundamental indicator for routing metrics [7], [5]. It is intuitive to believe that if two nodes contact more often in the past, they will be likely to meet in the future. Therefore in our prediction framework, CF will also be considered to measure the contact closeness of two nodes.

IV. Proposed Method

A. Problem Definition

We now introduce the general model for link pattern prediction. We firstly break the contacts in networks as a series of snapshots, represented by (G_1, G_2, G_3,$\cdots$, G_t). As shown in Figure. 2, each snapshot has an adjacent matrix E, which documents the contact times during the period of snapshot. Given a sequence of historical snapshots from G_1 to G_t, we want to predict the contact patterns of the next snapshot G_{t+1}. Actually the snapshots are the accumulated connections of certain duration, which is determined by the time granularity to predict.



$$E_1 = \begin{bmatrix} 0,2,0,0,0 \\ 2,0,0,0,0 \\ 0,0,0,1,0 \\ 0,0,1,0,1 \\ 0,0,0,1,0 \end{bmatrix} \quad E_2 = \begin{bmatrix} 0,3,1,0,1 \\ 3,0,0,0,1 \\ 1,0,0,2,0 \\ 0,0,2,0,0 \\ 1,1,0,0,0 \end{bmatrix} \quad E_3 = \begin{bmatrix} 0,1,1,0,0 \\ 1,0,1,0,0 \\ 1,1,0,0,0 \\ 0,0,0,0,2 \\ 0,0,0,2,0 \end{bmatrix}$$

Fig. 2. Network Partition

According to the prediction model, we also assume that the opportunity of establishing links in the future is determined by the current local network structure. That is to say, whether node 1 and 2 will establish a link in G_{t+1} is strongly influenced by their local network structure in G_t. This pair can be predicted if one step evolution of links can be captured. It is impossible to just infer from current snapshot so we need to learn from historical snapshots to get the probability of propagating a link between node 1 and 2. At last the links patterns are constructed through links with high probabilities.

B. Link Feature Set Definition

We first define the feature vector of two nodes (i and j) at timestep t.

$$f_t(i,j) = \{ls_t(i,j), cf_t(i,j)\} \tag{2}$$

where $f_t(i,j) \in F$, a finite feature set. Here $ls_t(i,j)$ and $cf_t(i,j)$ respectively represent the local score given by Eq. (1) and CF between i and j in snapshot G_t. Feature vectors in all cases can be found in the feature set F. Based on previous assumption that current network link pattern determine the future pattern, we assert that the node pairs with similar local structure will share the similar link probability in the next timestep. Feature vector $f_t(i,j)$ is a good representation for link structures around i and j at timestep t. If two pairs have $f_t(i,j)$=$f_t(i,j)$, they are likely to have the same probability to connect with each other in the next timestep. As a matter of fact, exact match of two feature vectors is impossible because ls in f is a fraction ranging from 0 to 1 after normalization. Two ls that have tiny difference will be regarded as the same value. So we make the approximation to replace the ls value in f with $[l \cdot ls]$, where l is an integer to limit the upper bound of replaced ls. According to point estimation, we now define link probability $p_{t+1}(i,j)$ of node pairs in next timestep $t+1$ to be:

$$p_t(i,j) = p(f_t(i,j)) = \frac{\beta_t^+(f)}{\beta_t(f)} \tag{3}$$

where$\beta_t(f)$ is the number of node pairs that have same feature f and $\beta_t^+(f)$ is the number of such pairs that have links in next timestep.

C. Estimation via Kernel Regression

We cannot get desirable results if we simply utilize Eq. (3) for link pattern prediction. Actually many of the statistical data points can bring a lot of noise that will finally disturb the actual link probability. For example, all pairs with the same feature contribute the same in Eq. (3) although some pairs have quite different local structure while others have very similar local structure. So we need to put weight to weaken some unrelated pairs and strengthen the highly related ones. Kernel regression can solve the problem quite well. It puts weight on similar links where more weight will be added if the historical data points are closer to the current data point. All of our work is based on the Nadaraya-Watson kernel regression [22] and the estimator is:

$$\hat{m}_h(x) = \frac{\sum_{i=1}^n K_h(x - X_i) Y_i}{\sum_{i=1}^n K_h(x - X_i)} \tag{4}$$

In this equation $\hat{m}_h(x)$ is the estimated y value of x. $\sum_{i=1}^n K_h(x - X_i)$ is the kernel function with a bandwidth h. This function defines the closeness between current data point (x, y) and the history point (X_i, Y_i) and the bandwidth h determines how smooth the estimating process will be. According to Eq. (4), our estimator of link probability is:

$$\hat{p}_t(i,j) = \frac{\sum_{x,y,t'} S(f_t(i,j), f_{t'}(x,y)) \cdot p_{t'+1}(x,y)}{\sum_{x,y,t'} S(f_t(i,j), f_{t'}(x,y))} \tag{5}$$

where $p_{t'+1}(x,y)$ is the probability of establishing links at timestep $t+1$ and $S(f_t(i,j), f_{t'}(x,y))$ measures the distance between two feature vectors. As a matter of fact, whether there exists a link between x and y is deterministic at time $t'+1$. Therefore $p_{t'+1}(x,y) = 1$ if x and y have links at time $t+1$, otherwise $p_{t'+1}(x,y) = 0$. In Eq. (5) all node pairs at any timestep are taken into consideration to measure their local similarity, which is low efficient. Taking all pairs into consideration does not equal to exact estimation because pairs with different feature from the current one will have little impact on estimation. In general, the closeness measurement in our model is:

$$\begin{aligned} S(f_t(i,j), f_{t'}(x,y)) &= K(\cdot) \cdot I_{f_t(i,j)}(f_{t'}(x,y)) \quad (6) \\ K(\cdot) &= K(M_t(i), M_{t'}(x)) \end{aligned}$$

where $K(M_t(i,j), M_{t'}(x,y))$ is the kernel function defined later. $I_{f_t(i,j)}(f_{t'}(x,y))$ is the indicator function to merely count pairs with same f, which greatly lowered the computation complexity by dropping irrelevant pairs. Combining Eq(5)(6), we can get:

$$\begin{aligned}\hat{p}_t(i,j) &= \frac{\sum_{x,t'} K(\cdot)\cdot\sum_y I_{f_t(i,j)}(f_{t'}(x,y))\cdot p_{t'+1}(x,y)}{\sum_{x,t'} K(\cdot)\cdot\sum_y I_{f_t(i,j)}(f_{t'}(x,y))} \\ &= \frac{\sum_{x,t'} K(\cdot)\cdot\beta^{+}_{xt'}(f_t(i,j))}{\sum_{x,t'} K(\cdot)\cdot\beta_{xt'}(f_t(i,j))} \end{aligned} \tag{7}$$

where $M_x(t') = \{\beta_{xt'}(f), \beta_{xt'}(f) \forall f \in F\}$ is a two dimensional matrix indexed by two values in feature vector f. Each element in $M_x(t')$ contains two values $\beta^{+}_{xt'}(f_t(i,j))$ and $\beta_{xt'}(f_t(i,j))$, which are similar to Eq. (3). But they are the number of such pairs in $x's$ local network at time t'. Actually in this equation we made the approximation that links with the same feature in the same local network have the same kernel despite minor differences in their surrounding connections.

D. Local Network Discovery

Now we focus on how to find the local networks for each node to construct the matrix . Two parts exist in $x's$ local networks $LN_t(x)$: its directed neighbor set at time t $N_t(x)$ and community $C_t(i)$ it belongs at time t. The algorithms for community partition into communities have been explored for years and some of them have been successfully applied in opportunistic networks [13]. But we do not intend to use them since they are time consuming when each snapshot is detected. Here we introduce a fast community detection method- AFOCS [23].

AFOCS is a dynamic community detection algorithm that can detect overlapping communities. This algorithm is divided into two steps. The first step is called FOCS, devoting to find the initial overlapping communities structure each node belongs. This will cost at most $O(n^2)$ time complexity. The next step is called AFOCS, where communities detected by FOCS are automatically updated when an edge is removed or added. This update process costs constant time so the complexity is up to the frequency of edge changes. That is to say the community detection is initially performed only once and maintained dynamically in the future. And the overlapping detected communities can enlarge scale of $LN_t(x)$ since the local networks are sparse in networks with few nodes.

E. Kernel Function

Kernel regression deals with numerical data but not the matrices so we need to define the distance between two matrices M_1 and $M_2(x)$. Each element in matrix M can compute the link probability $p(f)$ under feature f by $\beta^{+}(f)/\beta(f)$. But such point estimation does not represent the variation in the probability well. So we use the posterior distribution of $p(f)$ instead. It is a beta distribution with the expectation of $\beta^{+}(f)/\beta(f)$ because the appearance of link is a binomial distribution with parameter $p(f)$. Now the distance between two matrices M_1 and M_2:

$$D(M_1, M_2) = \sum_{f\in F} \delta(B_1(f), B_2(f)) \tag{8}$$

Algorithm 1: *finding neighboring matrices*

```
Input: $n$ sequences of $p^m < p^m_0, p^m_1, \cdots, p^m_{n-1} >$, the
        number of clusters k
Output: $k$ clusters and central point in each cluster.
//$\mu[k]$-central points and $c[n]$-cluster tags;
Initialization: int $\mu[k], c[n]$, i=0;
//$D(\cdot)$ is same with Eq. (11);
Distortion function: $J(c, \mu_k) = \sum_{i=1}^{n} D^2(x_i, \mu(c[i]))$ for
$i \le k; i++$ do
    $\mu[k]$ = Random($n$);
end
while $\Delta J < 0$ do
    for $i = 0; i < n; i++$ do
        for int $j = 0; j < k; j++$ do
            $c[i] = j$ if $i$ has the minimum distance with
            $\mu[j]$
        end
    end
    for $i = 0; i \le k; i++$ do
        update $\mu[i]$ by averaging cluster $i$'s sequences
    end
end
```

where $\delta(\cdot)$ is the total variation distance which measures the distance of two functions. $B_1(f)$ and $B_2(f)$ are beta functions indexed by f respectively from M_1 and $M_2(x)$. Then we select the Gaussian kernel function:

$$K(M_1, M_2) = exp\left(-\frac{D^2(M_1, M_2)}{2h^2}\right) \tag{9}$$

it has been proved by Scott [26] in Section 6.2.3 that the estimated results is insensitive to kernels but very sensitive to the bandwidth. We will minimize the cross validation score to find the optimal bandwidth h:

$$\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{p_i - \hat{p}_i}{K_i}\right)^2 \tag{10}$$

where p_i is the real probability of i, which is either 0 or 1 defined in Eq. (5). $\hat{p}_i$ is the estimator of Eq. (5) and K_i is its corresponding kernel weight. The cross validation score actually reflect the variance between real and estimated values. And it is a concave function where there exists a minimum value. We can approach the optimal bandwidth by enumeration.

F. Complexity

The basic idea to get the estimated value of pairwise probability in Eq. (7) is to numerate all history matrices. In view of the great expansion in matrices quantities, we have to find a approximate solution to deal with the time complexity. As a matter of fact, we do not need to numerate all possible points for estimation. We just have to find some neighbor points of the current estimated one like k-nearest neighbors(k-nn) estimation. In k-nn estimation, the estimated value is calculated by averaging all k nearest neighbors. The first step, however, is to find the k nearest neighbors with the cost of $O(n)$ time. This will not help in lowering time complexity in kernel estimation. As instead, we use the k-means clustering algorithm to find neighbors without limitation on the number of neighbors to count.

In our model, we aim at finding neighbors of matrix under the distance definition in Eq. (8). If we calculate the total variance by continuous Beta distribution, central point in each cluster cannot be calculated by average. We discretize the continuous Beta function to a sequence of probabilities. In this way, each matrix $M_i$can then be represented by a sequence of probability values p_i^m(m is the number of probability values) and the distance between M_i and M_j can be rewritten as:

$$D(M_i, M_j) = \frac{1}{2}\sum_{k=1}^{m} |p_i^k - p_j^k| \tag{11}$$

In Algorithm. 1, the input includes discrete probability sequences p^m and the number of clusters k. k roughly determines the number of neighbors that will be used for estimation. If k is too small, there will be few neighborhood matrices used for estimation and vice visa. After the execution of this algorithm, we can get the clusters and their central points.

k-means clustering is time complex with a cost of $O(nkt)$ where n is the number of points, k is the number of clusters and t is the number of iterations. In our PreKR framework, it was executed only at the initialization step. And for prediction in timesteps, each matrix have to find the nearest cluster by central point, where neighbors of this matrix can be found for estimation in Eq. 7. Also this matrix will be added to the corresponding cluster. Only after many timesteps intervals, the algorithm will be recalled for clustering adjustment.

V. Results

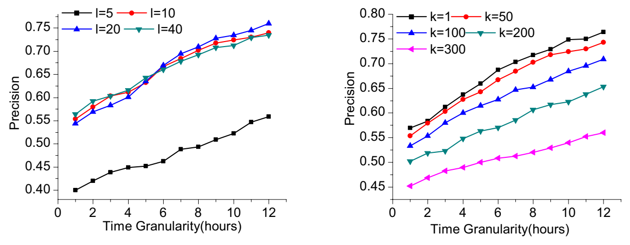
A. Datasets

In our experiment, we use three representative datasets: SIGCOMM [24], UCSD [21] and NetSense [19]. These three datasets are collected from either mobile devices or wireless access points and cover basic OppNets scenarios. Table I summarizes the datasets.

SIGCOMM. This dataset contains Bluetooth logs of 76 participants in 4 days. The connections in the first two days are especially dense for that most of the participants join main conferences in the first two days but went to other places for tourism in the last two days. Connections were especially sparse and invariant after conference dinner. So we just fetch 42 hours in the first two days for experiments.

UCSD. UCSD dataset has 276 users with PDA users of students collected in 77 days. Connections are established via WiFi access points and the device discovery interval is set to two minutes. This campus dataset is relatively in low density because activities are infrequently associated between students compared to conferences attendees. This dataset also have drawbacks that connections are random and unpredictable in the midnight since they are only affected by signal strength other than mobility. We take only 16 hours every day from 8a.m to 24p.m, like on and off periods in [6].

NetSense. NetSense was collected in Notre Dame University for about 15 months. Denser contacts are recorded in this dataset because users are forbidden to turn off Bluetooth and WiFi modules. The most significant advantage of NetSense dataset is its record on both Bluetooth and WiFi connections. This will help us compare reachability and predicting accuracy of Bluetooth and WiFi. Bluetooth device discovery interval is only 1 minute while WiFi remains 5 minutes to save energy. In this dataset, we select nearly one month data from November 2 to November 30 for evaluation.



(a) Choosing l (b) Choosing k

Fig. 3. Choice of Parameters in SIGCOMM

B. Experimental Setup

In our experiments, we use the Precision metric [20] to evaluate the accuracy of predicted link patterns:

$$Precision = r/s \tag{12}$$

where s is the number of links to be predicted and r is the number of links that was correctly predicted. We simply set s to be the actual links in the next timestep in PreKR performance evaluation.

C. Parameters Selection

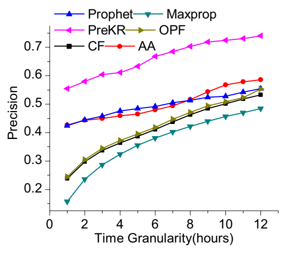
Generally we have to determine two parameters ahead of time: l used in Section IV-B and k used in Section IV-F. l decides not only the upper bound for adjusted ls but also the number of elements of each feature matrix. If l is larger, the adjusted ls will represent the original value better but more space are cost for matrix storage. As shown in Figure. 3(a), l slightly influences the accuracy of predicted pattern except $l = 5$. So we choose to use $l = 10$ for both accuracy and storage. As to the choice of k in Figure. 3(b), we find that prediction precision is insensitive to small k but very sensitive to large k. Estimation will be biased by large k because the number of neighbors found through k-means clustering are small. Therefore, we choose $k = 50$ for SIGCOMM dataset. Similarly, we choose $l = 10$ and $k = 200$ for UCSD dataset.

D. Accuracy Comparison

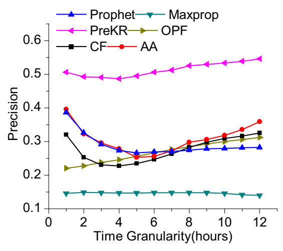
We compare PreKR with state-of-the-art link prediction methods. Apart from AA and CF used in PreKR, we also implement some predicting routing for comparison: Prophet [16], MaxProp [4] and OPF [17]. As can be seen in Figure. 4, PreKR outperforms other methods in both SIGCOMM and UCSD datasets. PreKR can perform up to 19% and 25% than

Traces	SIGCOMM	UCSD	NetSense
Collected Year	2009	2002	2012-2013
Devices	Smartphone	PDA	Smartphone
Module	Bluetooth	WiFi	Bluetooth & WiFi
Number of Devices	76	275	189
Duration(days)	4	77	458
Discovery Interval(min)	2	2	1 & 5
Environment	conference	campus	campus

TABLE I
THE DATASETS

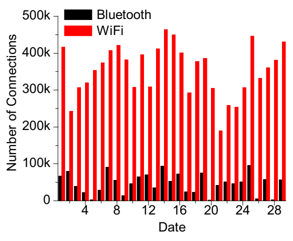


(a) SIGCOMM

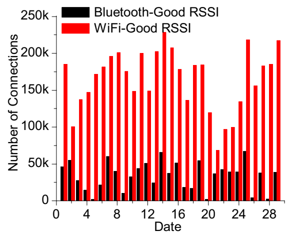


(b) UCSD

Fig. 4. Performance Evaluation



(a) All connections



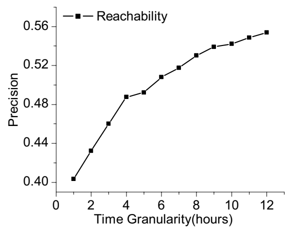
(a) Good connections

Fig. 6. Bluetooth and WiFi connection density

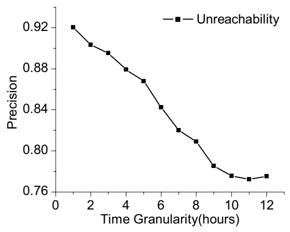
state-of-the-art methods in SIGCOMM and UCSD. Interestingly PreKR predicts better in SIGCOMM than that of UCSD, which may be caused by rough estimation of human mobility with Bluetooth connections.

E. Reachability and Unreachability Prediction

Actually 70% accuracy of predicting link patterns does not mean the 70% of the delivery reachability can be very well predicted. Reachable destination based on predicted patterns does not mean that they can be reached by real links. To verify the reachability of links, we use Epidemic routing to find real reachable destinations and search the predicted patterns to find whether they are reachable or not. As shown in Figure. 5, the accurate reachable of links is usually lower than the accuracy on predicted patterns, but the unreachability prediction is very successful. The mean accuracy of unreachability prediction can reach over 90%. This is very helpful for applications because it can be used to decide whether this this service should be denied or not. We saved more than 90% useless delivery although some of the packets sending are mistakenly denied.



(a)Reachability Accuracy



(b)Unreachability Accuracy

Fig. 5. Reachability and Unreachability Prediction in SIGCOMM

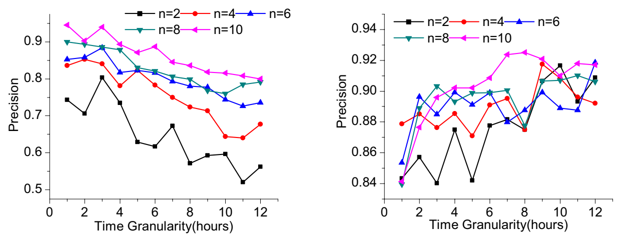
F. Bluetooth or WiFi

In OppNets layer, we investigate two types of communication: WiFiDirect and Bluetooth. The choice of physical design is dependent on demand of transmissions. The coverage of WiFi is at about 100 meters while Bluetooth can only reach 10 meters. Figure. 6 illustrates the connection density of both Bluetooth and WiFi in NetSense dataset. We get the number of links in everyday from Nov 2 to Nov 30 in 2012. In Figure. 6(a) we can see that WiFi connections are much more denser and has an average of 7 times of Bluetooth connections. In Figure. 6(b) the good RSSI refers to signal strengths larger than $-80dBm$, which indicates high transmission quality. The average connection frequency of WiFi is still 5 times that of Bluetooth for Good RSSI. The denser connections of WiFi brings a highly reachable networks.

However, WiFi is energy hungry in both device detection and transmission. According to results presented by Liu *et al.* [18], the energy consumption of WiFi is up to 3 times than Bluetooth. Now we compare the detailed PreKR performance on Bluetooth and WiFi in NetSense. In Figure. 8, we can see that the prediction of Bluetooth Connections is far better than WiFi Prediction, which also result in big difference in unreachability prediction. This is quite sound in that WiFi connections reflect mobility of users in much rough granularity while Bluetooth connections means a face-to-face meeting with each other. Better prediction performance for Bluetooth leads to more energy saving by efficient transmission. Therefore, our first choice is to utilize Bluetooth for opportunistic transmission. Then we try to use WiFiDirect if Bluetooth cannot satisfy transmission requirement.

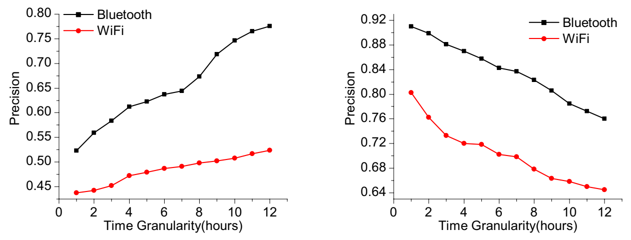
G. High Degree Nodes Prediction

In some OppNets applications, we have to predict and select high degree nodes to act as traffic centers for content



(a) SIGCOMM (b) NetSense Bluetooth

Fig. 7. High Degree Nodes Precision



(a)Link Pattern Accuracy (b)Unreachability Accuracy

Fig. 8. Bluetooth and WiFi Prediction Performance in NetSense

dissemination. Previous work select such nodes by either accumulation or average on historical degree, which could succeed in long-term prediction but fail in short-term prediction. We use link patterns predicted by PreKR to deal with short term prediction.

As shown in Figure. 7, we select nodes with n highest degrees in both predicted link patterns to find how many high degree nodes are correctly predicted. In SIGCOMM dataset, we can reach at least 52% to 94% accuracy. In NetSense Bluetooth dataset, we can reach 84% to 93%. And NetSense prediction has lower variation than that of SIGCOMM due to the more periodic daily campus life of NetSense participants.

VI. Deployment and Discussion

In this section we investigate the possible architecture for link pattern prediction deployment. We also show how our work can benefit prospective applications like mobile content sharing and mobile traffic offloading.

A. Heterogeneous Architecture

It seems impossible to deploy PreKR in OppNets since it collects all historical connections. But what if we deploy a centralized server and use the cellular networks (CellNets) for assistance like what has been discussed in [12]? This is two-fold advances:

- Deployment of centralized server relieves pains of energy waste by our PreKR framework and centralized control. The server executes all time and memory demanding tasks like link prediction , resource allocation and transmission decisions. Therefore mobile devices just need to perform packets transmission, whose failure risks have been decreased by PreKR.
- Deployment of CellNets relieves pains of waiting with no response by low delay transmission. Some packets are small but essential in OppNets transmission. We have to notify users service reachability and transmission progress, whose packets need real time transmission. Also, we prefer cellular traffic if only several kilobytes remains to be transmitted by OppNets.

As illustrated in Fiugre. 9, there are three layers in this heterogeneous architecture: OppNets layer, CellNets layers and Server layer. OppNets are responsible for large bulk of data transmission while cellular networks transmit small size but precious packets. The server layer predicts link patterns and makes transmission decisions to improve user experience for OppNets applications.

B. Delay Tolerant Applications

Recent years have seen great traffic demand for OppNets. In this section we investigate two prospective applications to demonstrate how our PreKR framework can improve user experience in these applications.

Mobile Content Sharing. In mobile content sharing systems, there are some delay tolerant content which can be transmitted via OppNets. For example If someone owns a film in his mobile devices and his best friends also want to watch it, will he pay it for tens of dollars or wait for several hours to get it? Most of the people will choose the later to save money. Such demand on delay tolerant content urges us to design OppNets based content sharing system like BlueTorrent [14]. In this system, they achieved short download time simply by finding optimal Bluetooth inquiry interval while deadly problems of no response and energy waste still cannot be solved. To this end, PreKR can help improve user experience by: 1) recommend users content that are highly reachable. 2) directly denying unreachable content request to save energy and storage.

Mobile traffic Offloading. In [19], experimental analysis on NetSense proved the feasibility of offloading tasks by OppNets. Han *et al.* [9] proposed a mobile traffic offloading system which utilizes OppNets to relieve the overloading problem in 3G networks. It designed algorithms to select highly connected nodes as target set to offload as much as traffic by OppNets. To this end, PreKR saves this step by: 1) predicting high degree nodes in predicted link patterns. 2) finding unreachable devices by OppNets and sending data directly by CellNets.

VII. Conclusion and Future Work

In this paper, we propose the framework for link pattern prediction in OppNets. We first choose the baseline indices that can represent the local network structure and then get the link probability estimation with kernel regression. Predicted link patterns are established by links that have highest probabilities. We exploit k-means algorithm to lower the complexity for kernel regression. Experimental results show that our method outperforms state-of-the-art methods. We also find that both unreachability and high degree node prediction can reach very

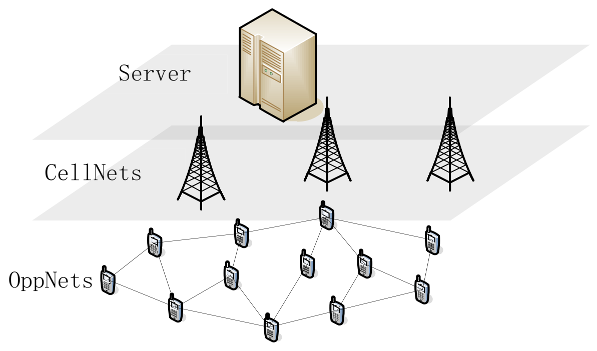


Fig. 9. Architecture for Opportunistic Transmission

good predicting precision in predicted link patterns. At last we explore heterogeneous system architecture for PreKR framework and show how PreKR could relieve OppNets problems in two different scenarios.

This paper, however, merely gives a rough estimation on future link patterns. We cannot predict individual link frequency and duration that are vital for traffic allocation. In our future work, we will devote to model individual links to predict link durations by linear time series analysis. We will design routing protocols based on both link patterns prediction and individual link prediction. To do more solid evaluation we will also implement a PreKR prototype system for mobile content sharing.

References

[1] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM06: Workshop on Link Analysis, Counterterrorism and Security*, 2006.

[2] A. Balasubramanian, B. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 373–384. ACM, 2007.

[3] C. Brouard, M. Szafranski, et al. Semi-supervised penalized output kernel regression for link prediction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 593–600, 2011.

[4] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *INFOCOM*, volume 6, pages 1–11, 2006.

[5] V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot. Diversity of forwarding paths in pocket switched networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 161–174. ACM, 2007.

[6] W. Gao and G. Cao. On exploiting transient contact patterns for data forwarding in delay tolerant networks. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 193–202. IEEE, 2010.

[7] W. Gao, Q. Li, B. Zhao, and G. Cao. Multicasting in delay tolerant networks: a social network perspective. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 299–308. ACM, 2009.

[8] R. Guimerà and M. Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.

[9] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan. Mobile data offloading through opportunistic communications and social participation. *Mobile Computing, IEEE Transactions on*, 11(5):821–834, 2012.

[10] D. Huang, S. Zhang, and Z. Chen. An optimal stopping decision method for routing in opportunistic networks. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 2074–2079. IEEE, 2013.

[11] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 10(11):1576–1589, 2011.

[12] P. Hui, A. Lindgren, and J. Crowcroft. Empirical evaluation of hybrid opportunistic networks. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1–10. IEEE, 2009.

[13] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, page 7. ACM, 2007.

[14] S. Jung, U. Lee, A. Chang, D.-K. Cho, and M. Gerla. Bluetorrent: Cooperative content sharing for bluetooth users. *Pervasive and Mobile Computing*, 3(6):609–634, 2007.

[15] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[16] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):19–20, 2003.

[17] C. Liu and J. Wu. An optimal probabilistic forwarding protocolin delay tolerant networks. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 105–114. ACM, 2009.

[18] S. Liu, Y. Jiang, and A. Striegel. Face-to-face proximity estimation using bluetooth on smartphones. 2013.

[19] S. Liu and A. D. Striegel. Exploring the potential in practice for opportunistic networks amongst smart mobile devices. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 315–326. ACM, 2013.

[20] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[21] M. McNett and G. M. Voelker. Access and mobility of wireless pda users. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(2):40–55, 2005.

[22] E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[23] N. P. Nguyen, T. N. Dinh, S. Tokala, and M. T. Thai. Overlapping communities in dynamic networks: their detection and mobile applications. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 85–96. ACM, 2011.

[24] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. Mobiclique: middleware for mobile social networking. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 49–54. ACM, 2009.

[25] A.-K. Pietilänen and C. Diot. Dissemination in opportunistic social networks: the role of temporal communities. In *Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*, pages 165–174. ACM, 2012.

[26] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*, volume 383. Wiley. com, 2009.

[27] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259. ACM, 2005.

[28] J. Su, J. Scott, P. Hui, E. Upton, M. H. Lim, C. Diot, J. Crowcroft, A. Goel, and E. de Lara. Haggle: Clean-slate networking for mobile devices. Technical report, Technical Report UCAM-CL-TR-680, University of Cambridge, Computer Laboratory, 2007.

[29] K. Xu, P. Hui, V. O. Li, J. Crowcroft, V. Latora, and P. Lio. Impact of altruism on opportunistic communications. In *Ubiquitous and Future Networks, 2009. ICUFN 2009. First International Conference on*, pages 153–158. IEEE, 2009.

[30] T. Zhou, L. Lü, and Y.-C. Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.