Chapter 1

Encounter-based Opportunistic Social Discovery in Mobile Networks

Udayan Kumar

Department of Computer and Information Science and Engineering, University of Florida, Gainesville, USA

Ahmed Helmy

Department of Computer and Information Science and Engineering, University of Florida, Gainesville, USA

CONTENTS

1.1 Introduction

Neighbor selection is a non-trivial problem faced in the design of opportunistic networks. Consider any opportunistic networking applications ranging from social discovery to P2P mobile games to DTNs and MANETs, almost all of the applications require neighbor discovery followed by some kind of neighbor selection. A DTN application may select a neighbor that has higher chances of relaying the message to the destination or a P2P mobile gaming application may select neighbor based on duration of past encounters. However, due to diversity of requirements, we argue that there is no one optimal method of selecting neighbors for all applications.

Selecting a subset of neighbors from all the available neighbors becomes challenging in the case of opportunistic network because 1. Neighbor selection is application dependent, 2. Identities of all the neighbors may not be known, 3. Users may not be comfortable communicating with unknown neighbors, 4. Without incentives, neighbors may not be willing to participate (mobile device are constrained for power & processing), 5. Not all neighbors may meet the requirement of the application, and 6. Some neighbors may have malicious intent.

At the same time, there are several unique characteristics of opportunistic networks that provide arsenal to tackle many of the above challenges such as 1. Physical proximity that enables easier verification of identity (one can have face-to-face meetings and also exchange out-of-band cryptographic keys), 2. Tight coupling of mobile devices and users can allow customization of selection based on user needs, and 3. Availability of location and other contextual information (e.g. mode of transport, importance of location) can give valuable insights when making neighbor selection. Face-to-face meeting, verification of user profile, and setup of out-of-band keys are comparatively low cost operations for neighbors in radio range as they are in physical proximity (for e.g. Bluetooth 4.0 range is < 50m) when compared to wired networks.

Utilizing the above mentioned characteristics, we provide a brief overview of studies in the areas of encounter based neighbor and social discovery, context awareness and recommendation (and reputation) systems for opportunistic network estab-

lishment. Furthermore, we present detailed discussion of the design and analysis of a new encounter-based framework, *ConnectEnc* (Connections based on Encounters) as a solution to the problem of neighbor selection. The framework is fully distributed, self-bootstrapping, privacy preserving, and integrates attack resilience mechanisms. This framework utilizes mobile encounters as a primitive to address the problem of neighbor selection. A mobile encounter signifies the detection of radio signals (Wi-Fi, Bluetooth, etc.) from another device (current neighbor). The use of short range radios (e.g., Bluetooth, Wi-Fi) enables detection and utilization of proximity and encounters. Furthermore, the tight coupling between users and mobile devices enables new and accurate ways to establish behavioral profiles that can be used to fine-tune the neighbor selections based on application requirements; e.g., by selecting say only the users encountered at multiple locations. Along with this encounter framework, we also promote the face-to-face interaction between peer to peer users that allows authentication peer identity and establishment of out-of-band encryption keys [14] that can be later used to establish secure P2P/opportunistic communication channel.

At the core of *ConnectEnc*, we use encounter rating metrics called *Encounter Filters*. These *Encounter Filters* analyze mobile encounters, proximity, location, and context data in novel ways, to augment the users (and application's) network/neighbor view and awareness. Its goal is to rate opportunities in terms of neighbors selections based on weighted filter scores that are coupled with the users input and application requirements. We investigate and detail five different algorithms applied to filter the encountered devices.

It is the fusion and integration of these multi-dimensional data, that provides the promise in selecting better neighbors in opportunistic networking in ways we could not before, and in ways that are not possible in wired networks due to lack of connectivity proximity. An opportunistic network application can now state its neighbor/peer selection criteria to the 'ConnectEnc' framework (such as neighbor with highest probability of meeting again or a neighbor who met at a particular location before, etc). The 'ConnectEnc' framework, based on *Encounter Filters*can provide the most suitable candidate(s) out of all the current neighbors.

This study introduces a systematic framework and new protocol for gathering and processing the encounter information to build encounter-based profiles of the neighbors. Evaluation of the 'ConnectEnc' framework and mobile application is a three-phase process: 1. real world mobile networks trace statistical analysis, 2. extensive trace-driven simulation of the framework components, and 3. prototype implementation and participatory testing on smartphones. First, we use wireless network traces from 3 different major university campuses spanning 9 months with over 70K users and 150 million encounters. We find that several filters possess desirable stability characteristics, and that selecting neighbors with high encounter scores in general forms a small world. Resilience to attacks (neighbors attempting to inflate encounter statistics), using anomaly detection, achieves less than 10% false positives and 7% false negatives. Second, we measure the effectiveness of *ConnectEnc* on epidemic routing in DTN with selfishness using neighbor recommendation by *ConnectEnc* and obtain higher network performance reducing the effects of selfishness. Third, we conduct a series of surveys and participatory experiments using *ConnectEnc*'s mobile

application to evaluate the performance of the framework against the ground truth. We find users' selection of trustworthy peers/neighbors (for opportunistic communication) has a statistically strong correlation with *ConnectEnc*'s recommendations. Further, *ConnectEnc* filters can capture 80% of the already known user within top 25% of the encountered users.

Key contributions of this work include: 1. introducing a framework to augment mobile user's perception and awareness of the network neighborhood by fusing multi-dimensional encounter and contextual data for better neighbor selection, 2. analyzing various trust adviser filters with extensive network traces, 3. propose a model for anomaly or attacker detection, 4. developing a mobile app 'ConnectEnc' that integrates the filters and contextual information to aid user in neighbor classification & selection, and 4. deployed *ConnectEnc* as a proof-of-concept mobile application to evaluate the framework based on ground truth via participatory testing.

1.2 Overview

This overview is organized into 4 different subsections, each corresponds to a step involved in establishing short-range-radio based mobile P2P networks. These 4 steps are i. Neighbor Discovery: here information for all the available devices is obtained, ii. Neighbor Selection: here a subset of all the available devices is selected, iii. Connection Establishment: after selection, peers exchange/negotiate connection parameters based on security and authentication, and iv. Applications: here we list some of the popular P2P applications.

1.2.1 Neighbor Discovery

In any P2P scenario, if the peers have unpredictable behavior (availability) in either space or time, there will be a need to discover peers that are currently available for interactions. Most of the popular P2P applications whether communicating over Internet or via Adhoc radio network employ some kind of P2P discovery mechanism. There are primarily two ways to discover i. using a central infrastructure (torrents) and ii. adhoc (sensor networks, DTNs). For opportunistic networking, the latter is more commonly utilized. To discover other peers, generally peers send out a discovery radio beacon to solicit response from all the neighboring peers. Several popular radio protocols such as Bluetooth and Wifi-Direct natively support this kind of discovery. Since a peer may be continuously moving (surrounding peers may also move), searching for available peers can be an expensive process in terms of energy consumption. Several energy efficient methods have been proposed (including one by the authors of this chapter) [35, 20]. There is also a research direction where based on the previous discovery patterns of a peer, predictions are made about future discovery of that peer [30]. Researchers find that human movement pattern is predictable at a coarse granularity based on that peer discoveries can also be predicted.

1.2.2 Neighbor Selection

Once a set of neighbors is discovered, there may be a need to select a subset of neighbors based on the requirements of the application. This step may be necessary when an application does not want to interact with all the available peers. The DTN routing application such as epidemic routing [34] may interact with all the available peers, however, several other DTN routing protocols such as [21, 26, 16, 36] may require selecting peers based on their encounter history. The main focus is on optimum end to end routing and less on one hop (next-hop) node selection. They may not be privacy preserving or may not provide stable recommendation and are not easily configurable to the needs of an application. Similarly, a gaming application may want to select peers who may be encountered again to finish off the game. Unlike neighbor discovery, there is no one way to select a peer. Different applications have different criteria.

The idea of neighbor selection in P2P networks has been well explored in wired networks where neighbors are selected based on the geographic proximity, latency, bandwidth available, etc. [27]. These ideas, however, do not hold when a mobile application wants to leverage a P2P based direct radio connection. Mobile P2P networks face greater set of challenges since the peers are mobile and there is a high possibility of peers moving out of radio range. There exist several DTN routing protocols that employ node selection algorithms [21, 26, 16] but focus mainly on optimum end to end routing and thus focus is less on one hop node selection. They may not be privacy preserving or may not provide stable recommendation and are not easily configurable to the needs of an application. The lack of any optimized one hop P2P neighbor selection is also a challenge for P2P mobile application development community [6]. There are several P2P applications available [5, 33, 4] but without any automatic strategy for peer selection, it is left out for the user to decide. But how will a user decide? *ConnectEnc* attempts to solve this problem by providing background information about the peers to make an informed selection. To best of our knowledge, there is no existing solution to this problem. Later in this chapter we propose, as a solution to this neighbor selection problem, a multi-criteria neighboring peer selection framework, *ConnectEnc*.

Several researchers have proposed novel approaches in peer selection using reputation based schemes, incentive based schemes, and game theory. The reputation based schemes target better peer selection based on previous interaction records by rating interactions with each peer. In [11], a node detects misbehavior locally by observation and use of second-hand information. In [10], a fully distributed reputation system is proposed that can cope with false information, where each node maintains a reputation rating for peers. In [31, 9, 15, 8], analysis of rewards provisions and punishment is conducted based on game theoretic approaches to provide incentives for message delivery. In [13], authors propose a game-theoretic model to discourage selfish behavior and stimulate cooperation by leveraging Nash equilibria with socially optimal behavior. In [38], authors propose a pricing mechanism to give credits to nodes that participate in the message forwarding mechanism. The cooperation is developed based on the number of messages transfered by the users.

A common theme in these works is the reliance on peer *interaction* to evolve the reputation/credit scores. Inherently, this creates an undesirable *circular dependence*, where interaction requires technology adoption (reputation/credit system), which - in turn - requires trust in specific instances of these systems. Hence, there is a compelling need for a *bootstrap* mechanism, which we directly address in our proposed design. Further, unlike other studies, we do utilize encounter context in this paper. Our work contributes towards solving this challenge by providing inputs from user's location preferences and contextual (e.g., social) behavior.

1.2.3 Connection Establishment

Once a neighbor is selected, the next thing to do is to establish a connection with the neighbor to enable data/information exchange. The connection establishment involves several challenges mostly from the security and privacy standpoint. Challenges such establishing secure connection between the nodes and identifying a peer when meeting again are some of the bigger challenges faced by mobile P2P connection establishment. Authors of [25, 14, 28] propose explicit authentication mechanism to generate trust and cooperation in network. These approaches are better modeled for small groups [25] and require exchange of public keys and the installation of the private key on the users device [14]. Another step to secure a connection can be to meet the peer/neighbor face-to-face (since radio-range of mobile devices is limited, peers must be co-located physically), verify the peer and can also setup out-of-band encryption keys. The out-of-band encryptions keys can be setup by the peers simply by exchanging a secret code word when meeting face-to-face or can also achieve cryptographic strength by using [14].

1.2.4 Applications

Whether it is a DTN application or a mobile P2P application, once a connection is established, these applications can start leveraging the established connection. A few examples of existing P2P applications are P2P multiplayer mobile gaming [5], cooperative sensing [23], mobile proxy [4], social discovery [3], personal safety [33], and Cellular offloading [19]. This generation of applications do not employ a neighbor selection method, hence they are mainly human driven in terms of neighbor selection; with very minimal or no automated sensing and selection of neighbors.

In the following sections we present our proposed *ConnectEnc* framework that can automate neighbor selection process by providing automatic requirement-specific neighbor selection. The framework provides novel ways to rate the neighbors and integrates within itself existing peer-rating systems such as recommendation and reputation systems. We begin with the rational for the design and then proceed towards design principles. Following the design, we present details of *Encounter Filters*, comprehensive analysis of the framework and a section on validation, along with a summary of *ConnectEnc* user study.

1.3 Rationale & Architecture

In this section, we present rational, design goals and high level design of the *ConnectEnc* framework.

1.3.1 Rationale and Approach

P2P mobile application can be only used with peers who are in the radio range. In cases where several peers may be around to participate in a P2P activity, which out of those should be picked? For some application any set of peers would work but many applications as mentioned earlier would require an informed selection based on the requirements of the application. For e.g. which peer has higher chance of encountering again or of a longer encounter session. Also users may not want to interact with randomly selected peers (Sec. 1.6.1).

Looking at this problem from a user's perspective, who will be the peers this user may meet again? (or other longer duration, etc) These will be the peers who are similar to user in their behavior (being at the same place and the same time). In social science this is known as principle of Homophily [29]. Homophily can be measured is several different ways and using encounter history we can measure spatio-temporal homophily. We propose several *Encounter Filters* to measure spatio-temporal homophily. For greater trust and reliability (optional) on a peer, a user can meet this peer face-to-face. This is easier in mobile P2P network as the peers are in physical proximity and some peers may already be socially known (although we do not make any such assumptions). Face-to-face meeting can be utilized to verify the authenticity of peer's claim and can be used to set up out-of-band encryption keys [14, 25]. If these keys are stored with *ConnectEnc* then other applications can use this key to securely communicate with the specific peer.

1.3.2 Design Goals

The main design goals for *ConnectEnc* include:

1. Balanced Discovery: In our peer selection (and discovery) framework, identifying peers known to the user (i.e., a perfect matches) is not always our goal. Instead, we aim to provide the user with a balance between acquaintances and new matches as a more useful and realistic measure. We achieve this by generating encounter scores over several filters and allowing application-specific peer selection.

2. Stability: The peer recommendation should be stable over time and insensitive to minor, temporary changes and noise in user behavior. Outliers and anomalies should be detected and removed.

3. Distributed Operation: *ConnectEnc* should be able to provide all the functionalities in a distributed fashion without the need for a centralized infrastructure or trusted third party. All operations should be performed locally on the users device. Not sharing of user information should be required by the system for privacy preservation.

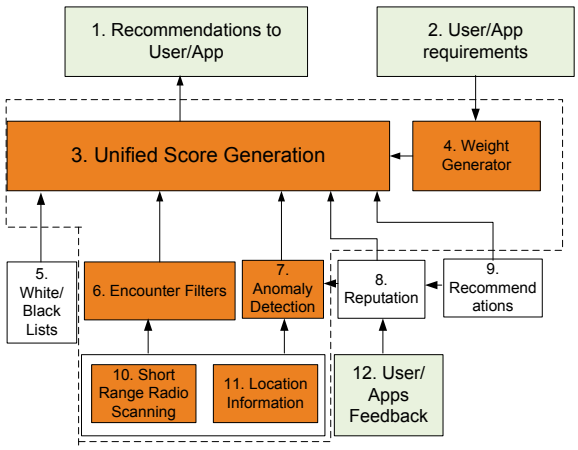


Figure 1.1: Block Diagram overview of the *ConnectEnc* architecture. Dotted lines enclose the modules of *ConnectEnc* (Orange colored blocks). Green colored blocks (1, 2 & 12) illustrates the blocks that interact with other applications and user of the device. White colored blocks (5, 8 & 9) illustrate the integration of external systems with *ConnectEnc*.

Other goals include: resilience (against attacks), power efficiency, and flexibility to utilize external sources (reputation & recommendation).

1.3.3 Overall Design

An architectural overview of the *ConnectEnc* framework and its related subsystems is provided in Fig. 1.1. Overall there are 3 categories of blocks: 1. Orange colored blocks (3, 4, 6, 7, 10 & 11) indicating the core components of *ConnectEnc*, 2. Green colored blocks (1, 2 & 12) indicating the modules that interact with the applications and users, and 3. White colored blocks (5, 8 & 9) indicate examples of external systems that can be integrated with *ConnectEnc*.

All the core components of *ConnectEnc* are fundamental to the design of the framework. These modules are required to meet the design goals. The basic functionality of each of the modules is as follows; The *Short Range Radio Scanning* module provides basic encounter information (for e.g., Bluetooth, WiFi AP discovery). The *Location Information* module provides the device's positioning data. This data is now received by *Encounter Filters* and *Anomaly Detection* modules. The *Encounter Filters* is the block that generates encounter scores using a family of filters (described in the next section). The *Anomaly Detection* provides a recommendation regarding suspicious encounter activities. The *Unified Score Generation* module combines the output of *Encounter Filters* with the output from *anomaly detection*, *recommendation system*, *reputation system*, and *black and white lists* using the weights provided by the *Weight Generator*. The *Weight Generator* provides weights that decide how

much importance is to be given to the different inputs to *Unified Score Generation*. The selection of weights is done based on the requirements given by the application.

The green colored blocks in Fig. 1.1 indicates how applications and user can interact with the *ConnectEnc* framework. We perceive the applications will interact with *ConnectEnc* framework by first setting up the requirements by specifying either relatively or absolutely the importance (weight) of each input considered by the *Unified Score Generation*. Once the weights are selected, *ConnectEnc* will generate a ranked ordered list of the encountered peers (Block 1) in the neighborhood. Once the application finishes the transactions with the neighboring devices, it provides (optional) feedback about the experience with the users. This feedback is feed into the *Reputation* block.

The white colored blocks are optional and external components of the *ConnectEnc* framework. These modules can enrich the peer selection process but are not required. Any existing systems providing necessary functionality can easily be integrated with this framework. The 'Reputation' block receives peer feedback from applications based on application's experience with this peer device. The 'Recommendation' block runs an external recommendation service and provides input to the framework. The 'White/Black List' allows users to explicitly give score to a device. This can empower user to add peers without even encountering them.

With this conceptual understanding of the system, we now describe the heart of *ConnectEnc* framework, *Encounter Filters*.

1.4 Encounter Filters

Encounter Filters rate encounters in multiple dimensions so that applications and users can make a selections based on rich set of choices. Due to lack of space we are going to discuss and analyze 5 major filters, however, the design of *ConnectEnc* is modular and can easily integrate more filters (if needed). The filters we propose and investigate are based on: *i.* Simple encounter (frequency and duration) ranking and *ii.* Spatial Correspondence.

1.4.1 Simple Encounter Ranking

These filters rate encounters by aggregating the encounter data using simple statistics. They are:

Frequency of Encounters (*FE*): ranks encountered devices based on total number of encounters over a window of history, regardless of the duration. So if a peer *A* is encountered more number of times than peer *B*, peer *A* will get a higher rank than *B*. For an encounter session (continuous uninterrupted encounters) FE score for the peer is increased only once by one. This filter score can be useful for applications when they have to decide between peers based on the chances of meeting again. Simply put higher FE score means higher chances of meeting.

Duration of Encounters (*DE*): ranks encountered devices based on the duration

of encounters. Encounter duration can be measured in two ways: i. total duration of encounters, ii. average session duration per encounter. An application using this metrics to decide between peers will find that higher DE score would mean that peer may have higher chances of having longer duration encounter session. Through our trace based analysis we find that both measures of DE have a statistically strong correlation with each other. Since, the first measure requires less storage and computation, we choose its score to represent DE score.

1.4.2 Spatial Correspondence

Spatial Correspondence based measures rate encounters on similar location visitations patterns. Higher spatial correspondence means that the peer is very similar in visiting locations as the user herself. Selecting a user with higher spatial correspondence means selecting a user who may be encountered more at locations preferred by the user. Spatial Correspondence can be measure in multiple ways, we present some of those techniques below.

Profile Vector (*PV*): To capture spatial correspondence, we have designed PV filter that stores location visitations of a user in a single dimensional vector. It is assumed for this filter that a device has some localization capability, which is quite common for today's devices. Each device maintains a vector. The columns of the vectors represent the different locations visited by a user and the values stored in each cell indicate either duration or count of the sessions at that particular location. At each location visit, the vector is updated with respect to the location.

To get encounter score, this vector is exchanged with other user and the inner product of the two vectors is computed. This score is higher if the two *PV*s are similar and can be zero, if the users do not have any visited location in common. Here, implicit weight is given to locations based on the count/duration spend. We can also provide an option to the user, where locations can have explicit weights.

However, this filter is not privacy preserving and can introduce attacks in the system, where a peer can tamper with its vector, also there are communication costs involved in exchanging the vectors. This problem in solved by *LV* filters at cost of having lesser information to compute similarity scores with.

Location Vector (*LV*): *LV* filter is very similar to *PV*, except that a user not only maintains a vector for itself but also for each encountered peer. The columns of the vectors represent the different locations visited by a user and the values stored in each cell indicate either duration (*LV-D*) or count (*LV-C*) of the sessions at that particular location. For every encounter, the vector for the encountering peer is updated with respect to the encounter location. Illustration in Fig. 1.2.

Since vectors for all the encountering peer are maintained locally on the device, LV requires no exchange of vectors among users for calculating similarity. This is more privacy-preserving and more resilient to attacks since only first-hand information is used (equivalent to what user might have observed). This privacy comes at the cost of requiring extra storage space for storing vectors for each user. Considerable storage optimization is achieved by storing (for each encountering user) only the locations where encounters happened. Similarity calculations are similar to PV.

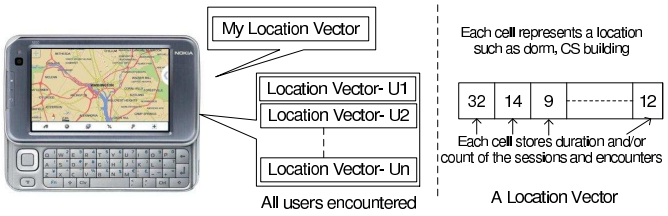


Figure 1.2: Location Vector *LV* for a user

Behavior Matrix (*BM*): The behavior matrix captures a spatio-temporal representation of user behavior. Columns of the behavior matrix denote locations and rows represent time units (days in our case). The value stored at each cell is a fraction of the on-line time spent by the user at a particular location on a particular day (see Fig. 1.3). Each user maintains their own matrix. To get the correspondence score, users can exchange and compare the two matrices.

To make the behavior similarity check efficient (in terms of space and computation complexity) and privacy preserving (as only the summary of matrix is exchanged), we use the eigen values of the behavior matrix for exchange between the two users. The eigenvalues are generated using SVD (Singular Value Decomposition). SVD is applied to a behavior matrix M, such that:

$$M = U \cdot \Sigma \cdot V^T, \tag{1.1}$$

where a set of *eigen-behavior* vectors, $v_1, v_2, \ldots, v_{rank(M)}$ that summarize the important trends in the original matrix M can be obtained from matrix V, with their corresponding weights, $w_{v_1}, w_{v_2}, \ldots, w_{v_{rank(V)}}$ calculated from the eigen-values in the matrix Σ. This set of vectors is referred to as the *behavioral profile* of the particular user, denoted as $BP(M)$, as they summarize the important trends in user M's behavioral pattern. The *behavioral similarity* metric between two users' association matrices A and B is defined based on their *behavioral profiles*, vectors a_i's and b_j's and the corresponding weights, as follows:

$$Sim(BP(A), BP(B)) = \sum_{i=1}^{rank(A)} \sum_{j=1}^{rank(B)} w_{a_i} w_{b_j} |a_i \cdot b_j| \tag{1.2}$$

which is essentially the weighted cosine inner product between the two sets of *eigen-behavior* vectors.

BM, like *PV*, is not privacy preserving, but can provide better spatio-temporal similarity calculations. Due to its privacy preservation, in the following sections, we have only used LV filter for spatial correspondence.

1.4.3 Hybrid Filter (HF)

Each filter provides a different perspective on an encounter or behavioral aspect. The hybrid filter provides a systematic and flexible mechanism to combine the scores

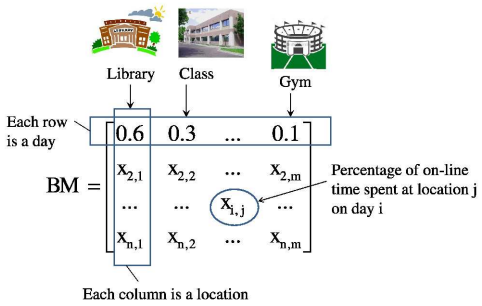


Figure 1.3: Behavior Matrix for a user

from all filters and present a unified score to the users. The selection of weights for various filters would depend on several factors including user's preference and feedback (check Sec. 1.6.1) and application requirements. A generic Hybrid Filter score (H) for a user U_j can be generated by using the following:

$$H(U_j) = \sum_{i}^{n} \alpha_i F_i(U j) \tag{1.3}$$

where $F_i(U_j)$ is the normalized score for user U_j according to filter i. The α_i is the weight given to filter score F_i and n is the total number of filters used. We select α_i such that $\sum \alpha_i = 1$, and $0 \leq \alpha_i \leq 1$.

This linear combination is chosen for its simplicity[1]. Our implementation allows users to customize these weights. From the analysis of user feedback (Sec. 1.6.2), we find that not all the users prefer same weights.

The processing and storage overheads for all the filters are shown in Tab. 1.1.

1.4.4 Decay of Filter Scores

Users may have a change in lifestyle (e.g. move to a different city, switch jobs) and may not very often encounter some of the previously highly rated peers. So, there may be a need to decay the score of peers, if they have not been encountered in a while. To design the decay of encounter scores, we borrow from social science studies that have shown that social relationship are dynamic and require frequent interactions to prevent decay. The strength of relationship wanes with the increase in time between interactions. This decay follows a exponential decay pattern with half time dependent on the relationship type [12] (3.5 years for family, 6 months for colleagues). We use a similar function to decay the filter scores with a user configurable half-time with 6 months set as default.

[1] Other non-linear combinations shall be investigated in future work.

Filter	Processing Overhead	Storage Overhead
FE	$O(m)$	$O(n)$
DE	$O(m)$	$O(n)$
PV	$O(m)$	$O(l)$
LV	$O(m)$	$O(nl)$
BM	$O(m)$	$O(ld^2)$ for SVD
HF	$O(n)$	$O(n)$

Table 1.1: Overhead of Filters in terms of processing and storage. Here m is the total no. of records in the encounter file, n is the no. of unique encountered user, l is no. of locations visited d represents the no. of days used for BM calculations. We also assume that $m >> n$.

1.5 Trace Based Analysis

To evaluate our design of Encounter Filters, we consider anonymzied trace sets from three universities (see Tab. 1.2; the information provided in the traces is anonymized; name of University U1 is also anonymized). The advantage of using WLAN traces is that they are much closer to reality in terms of user mobility (also representative of a larger population) than the existing synthetic mobility models. However, due to lack of ground truth in WLAN traces, we also collected traces with ground truth by deployment of *ConnectEnc*[2]. The results from the deployment are discussed after this section. The WLAN traces, much like other real traces, have small percentage of noise and error. We assume that users associating to same wireless Access Point (AP) encounter each other as AP range is generally less than 50 meters indoors and most of the traces are from indoor usage. It is assumed that each unique device (identified by MAC address) represents a user.

We use the WLAN traces to generate Encounter Filter score for each user found in the trace. The WLAN trace is converted to encounter trace for each user by determining all the other users who had overlapping sessions with this user at the same AP (location). Encounter Filters take this encounter trace as an input and produce a ranked list by encounter score. For analysis, we pick top T% peers of a user from the ranked list. We investigate three properties of the filters: 1. Correlation among filter, 2. Stability, and 3. Small world characteristics.

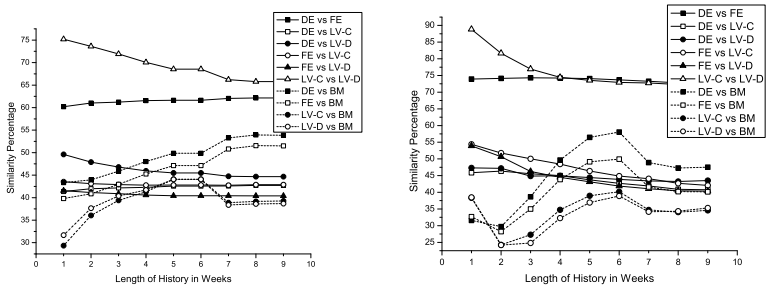
1.5.1 Filter Correlation

We examine the degree of similarity (correlation) among scores from different filters. While high similarity indicates redundancy of the filters, low similarity implies orthogonality of the recommendations. For this investigation, we have considered 9

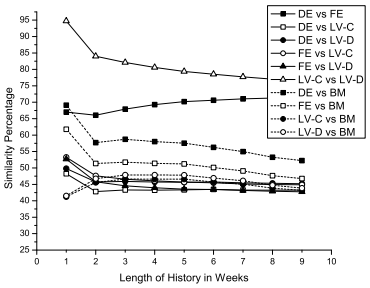
[2] MIT Reality Mining [17] traces have ground truth in terms of survey data. However, the average number friends per person is close to 1 (including several users who have listed themselves as their friends). Therefore, this trace set cannot be meaningfully used for evaluations.

Trace Source	U1	USC [22]	Dartmouth [1]
Time/duration of trace	Fall 2007	Spring 2007	Fall 2005
Start/End time	09/01/07-11/30/07	01/01/07-03/30/07	09/01/05-11/30/05
Unique Locations	845 APs	137 buildings	133 APs
Unique MACs analyzed	34694	32084	4906

Table 1.2: Facts about studied traces

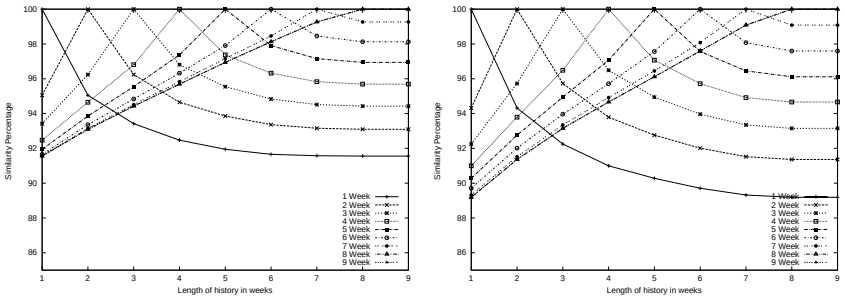


A. U1 B. Dartmouth

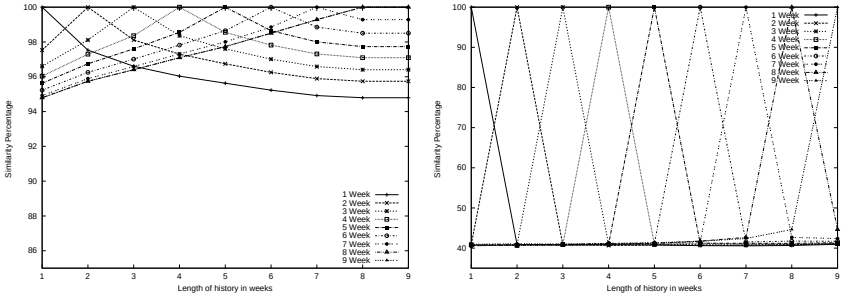


C. USC

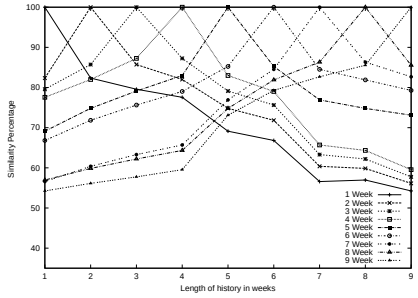
Figure 1.4: Correlation between the encounter lists produced by various filters at threshold, T=40%



A. Duration of Encounter(DE) B. Frequency of Encounter (FE)

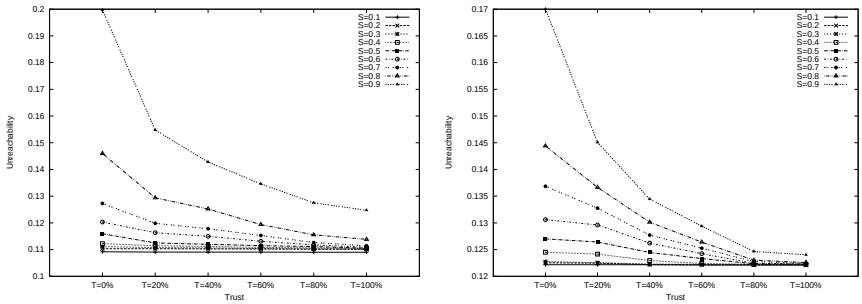


C. Location Vector - Count($LV-C$) D. Location Vector - Duration ($LV-D$)

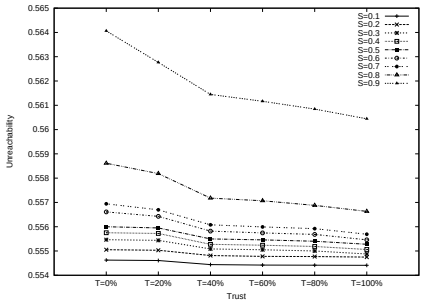


E. Behavior Matrix (BM)

Figure 1.5: Comparison of encounter score list belonging to different history for various filters at T=40% (note that the y-axis scale for DE starts at 85% and for $LV-D$ and BM the scale starts at 35%).



A. U1 B. Dartmouth



C. USC

Figure 1.6: Average unreachability with varying encounter score threshold - *T* and selfishness - *S* using DE filter

week long traces and threshold the score list at $T = 40\%$ for varying length (at 1 week interval) of encounter history (results for other T values show similar trend).

As Fig. 1.4 shows, the trends are similar across the traces. $LV - D$ and $LV - C$ filter results show $\sim$70% similarity as the list stabilize around 9 weeks of history. FE v.s. DE stabilize around 60% to 70%. Rest of the filters stabilize between 55% to 30%, meaning they produce different sets of lists. The low similarity indicates that filters are not redundant and can be used to generate rich set of recommendations.

1.5.2 Filter Stability

When an application requests node recommendation, giving the criterion for selection, it may want to know if this recommendation will hold true in future. For e.g. will a peer who had frequent encounters in the past, maintains a similar trend in future (user is assumed to maintain same lifestyle). Basically, are *ConnectEnc*'s recommendations stable in time? Moreover instability can confuse users and reduce the effectiveness of in-application cache. Therefore, it is imperative to examine stability in the peer recommendation over time. We investigate the stability of the peer lists at $T = 40\%$ using 9 weeks of U1 traces (other T values and traces show similar trend). Peer lists from multiple trace lengths are used to examine stability.

More than 90% similarity is found between 1 and 9 weeks trace for DE, FE and LV-C filters (see Fig. 1.5), implying that users selected in 1st week of encounter continued to be in the peer recommendation list of 9 week long encounter history. BM filter shows high stability when the difference in history is less than 2 weeks (80%) and falls to 55% for 1 week and 9 weeks. The LV-D filter shows similarity of about 40% between any list, implying that every week the list changes by 60%. This indicates that users may encounter regularly (by stability in LV-C) but may spend different amount of time encountering over the weeks. Overall, we note that some filters (DE, FE, and LV-C) stabilize in just 1 week of history, which makes them suitable for recommendations when encounter history is short. The time interval between the recommendation list regeneration can also be long (reducing processing requirements).

1.5.3 Graph Analysis

We analyzed the effect of peer recommendations on the network graph and compared it with the regular and random graphs while increasing selection threshold (T)(using DE filter, other filters show similar results). An edge is added between a pair of nodes only when atleast one of them is peer recommended by each other (un-directed graph). We note that clustering coefficient (CC) [7] of the network increases with T% and the path length (PL) decreases with increase in T%. For e.g., using 9 week U1 trace, CC is 0.171 at $T = 10\%$ and becomes 0.201 at $T = 100\%$. However, in the same scenario Path Length decreases from 3.64 to 2.59. More than 99% of the nodes were connected even at $T = 10\%$.

A small world analysis is performed as described in [7]. We find that normalized CC (NCC) is close to CC of regular graph and the normalized PL (NPL) is close to

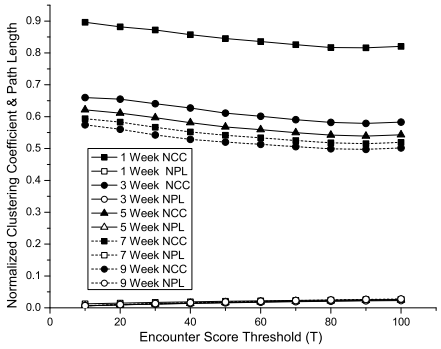


Figure 1.7: Normalized Clustering Coefficient and Path Length

PL of the random graph (Fig. 1.7 shows NCC and NPL for different lengths of traces and values of T). It appears that network created by peer recommendations is a small world network (results for other traces are similar).

1.6 Implementation & Simulation

In this section we show our validation of three major questions regarding the design of *ConnectEnc*: i. Do people prefer connecting with peers they already have some information on, ii. Is *ConnectEnc* able to discover peers that users may want to connect to?, and iii. Can *ConnectEnc* recommendations be useful in a P2P communication scenario? First point is to check the premise of our assumption that a user may have preferences in selecting a peer that in a way can affect how application select neighbors (user may add constraints such as I only want to play this game when there are higher chances of finishing this game later). We tackle this question with a survey. The second point is to validate that if users prefer selecting peers who have higher encounter score, is *ConnectEnc* able to discover them? We perform a user study using *ConnectEnc* mobile application to address this question. For the third point, we take DTN routing as our P2P application. We show, with the help of large-scale trace-driven simulations that *ConnectEnc* recommendations can lead to better routing in DTN networks having selfish nodes.

1.6.1 Survey

To investigate whether people prefer connecting with peers they already have some information on, we conducted a survey at a major computer network conference, this population has good understanding of computer networks. Participants were asked to indicate their willingness to communicate (using P2P applications) under different scenarios on a scale of 1 to 10. We received 32 usable responses. As Fig. 1.8 shows, willingness of the users to cooperate with unknown user/device is low (mean is 2.31).

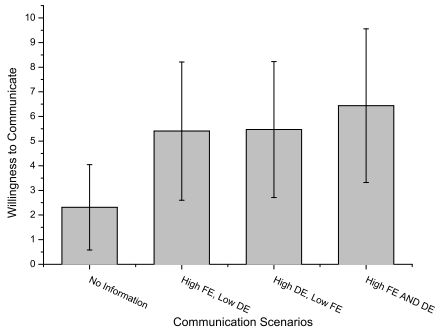


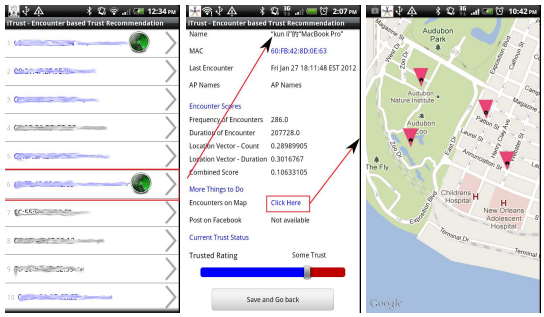
Figure 1.8: Survey Results showing user's propensity to communicate with other users in various communication scenarios

However, willingness increases when users have knowledge about the encounter history. This reinforces the approach of *ConnectEnc* of using encounters to make peer selection. We also observe that users give more importance to combined scores (FE and DE score are high) than individual scores (FE is high or DE is high). This justifies *ConnectEnc*'s use of Hybrid Filter for combining encounter scores. Standard deviations in results suggest that although most users want information about encountered users before cooperating, the individual importance of the filters may vary. This flexibility is made available in *ConnectEnc*'s Hybrid Filter (more generically by Unified Score Generation) by assigning weights according to user's preference.

1.6.2 ConnectEnc Application

To investigate whether *ConnectEnc* is able to discover peers that users may want to connect to, we developed a *ConnectEnc* mobile application and conducted a user study. The application measures the mobile encounters (over Bluetooth radio) and rates the peer devices based on the score of *Encounter Filters*. The application allows user to mark a device as trusted if they would like to have any P2P communication with that device in the future. We collect this selection data and correlate the user selections with *ConnectEnc* recommendations (based on encounter score) to validate our approach.

Currently, *ConnectEnc* is available for Android platform and Linux based Nokia Tablet N810 [2]. It provides the ability to rate encounter users based on FE, DE, LV and Hybrid filters. Encountered users can be sorted by any filter and weights for the Hybrid filters are user configurable. If some of the encountered users are currently discoverable, their listing would have a green circular mark as shown in Fig. 1.9A. The application provides inbuilt facilities for scanning Bluetooth devices and Wireless Access Points (for localization as GPS is energy-wise expensive. User can select GPS, if needed). On selecting a particular user, encounter details (Fig. 1.9B are presented and clicking on the map option one can see encounter locations on map (Fig. 1.9C). Apart from the filter scores, other statistics such as distribution



A B C

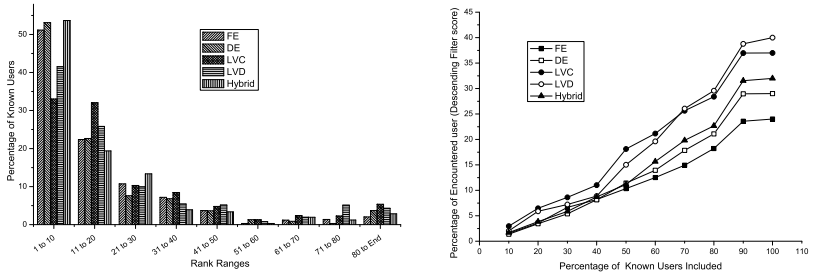
Figure 1.9: Selected screenshots of *ConnectEnc* application (earlier it was named iTrust). Fig. A. shows the main screen where encounter users are sorted by the filter score (Names and MAC are blurred intentionally). Current encounters marked with Green circles. Marked known users are shown in Blue color. Fig. B. shows details for an encountered user. Fig. C. shows user encounters on Map. Red colored annotation is added to show the application flow

of encounters with a peer over time are also available. Encountering devices can be rated for trust (P2P communication oriented) by the user on the scale from -2 (no Trust) to 2 (high Trust). This application is also capable of providing peer selection information to other applications. This application can also be used a social discovery application, where it can alert user about neighboring peer devices and give context by showing history and location of past encounters. We note that use of *ConnectEnc* does not affect privacy of the users. *ConnectEnc* only stores information on discoverable Bluetooth devices. Any Bluetooth capable device can capture the same information that *ConnectEnc* captures.

Application Evaluation: 22 students (grad and undergrad) from CS major ran *ConnectEnc* app for atleast a month. Users were asked to mark devices they trust (for P2P communication) in the application. On average, the number of trusted peer marked by each user is 15 and the number of unique devices encountered per user is 175. We use this data to investigate if recommendation by *Encounter Filters* correlates with trusted user identification. We note that not all encountered users who may be trusted/non-trusted may have been marked and not all trusted users may have discoverable Bluetooth. This issue will be of lesser concern as the adoption of *ConnectEnc* increases.

We rated the performance of *ConnectEnc* for each of the 5 filters (including Hybrid Filter, referred as Combined Filter (CF) in the app, with equal weights) on 2 metrics, 1: number of trusted peers in range top 1 to 10, 11 to 20, etc of *ConnectEnc* recommendations (also known as Precision metric in Information Retrieval literature) and 2. fraction of encounter peers needed (from top) to capture 'x'% of trusted peers for each filter. The above metrics are chosen to measure how well the filters perform when compared to user's selection. Here ranking is based on the filter score.

For metric 1, we note that *ConnectEnc* is able give high ranks to trusted peers



A. B.

Figure 1.10: ***ConnectEnc*** **evaluations based on application usage. Fig. A shows the percentage of trusted users in 1 to 10 Top user, 11 to 20 Top users for each filter. Fig B. shows fraction of encounter users needed (from top) to capture 'x'% of trusted users for each filter**

(Fig. 1.10A.). On average, out of top 10 ranked peers recommended by FE, DE and CF, 5 (50%) or more peers are marked trusted. We see that LV filter's top 10 ranks have 3 to 4 peers on average, however, if we consider top 20 peers, all filters capture 6-8 trusted peers (more than 50% of the total trusted peers). The number of trusted peers in rest of the ranges continue to fall except in the last range as it contains all the peers ranked beyond 80. For all the filters, there is a strong statistically significant correlation between the score and the rank of trusted peers (e.g., for LVC, r=0.84, p $<$0.01). Evaluations using metrics 2 shows that 80% of the trusted peers are captured by top 25% of the encountering peers as ranked by the filters and their is a strong statically significant correlation (Fig. 1.10B.). This shows users willingness to trust others (for P2P communicaton) in a mobile network to statistically correlate with recommendation given by *ConnectEnc*. We also note that there are peers who have high rank, yet they are not trusted. We believe, these can be the encountered peers, who are very similar to the user and can provide new interaction opportunities to the user and can be utilized by other mobile applications (including social networks).

Another finding from the deployment is that average storage requirement for *ConnectEnc* to store one month of data is 6.2MB including raw and processed data (75MB per year). This implies that with the current availability of mobile devices with multi GB storage capacity, ConnectEnc's storage requirements can easily be met. We have also used this deployment data to create an energy efficient encounter scanner as explained below.

Energy Efficiency: Scanning of Bluetooth and WiFi devices consumes considerable power (since the scanning process is periodic). After receiving the traces (which were scanned at 1 min interval), we noted that due to spatial locality in the traces, we can skip the scanning rounds if we find the same devices again in the next round, assuming that the user remains in the same location. The number of rounds we skip is $(2^n - 1)$, where n is number of times same devices are found consecutively, with

an upper threshold (MaxThres). If after a scan round, the devices change, we make $n = 0$. We note that reducing scanning period increases the loss of encounter information. Since we have the ground truth (traces scanned at 1 min), we can find out the information lost using L1 norm on the distribution of AP (Wifi trace) and Bluetooth devices in both the cases. We note that $n = 2$, gives us 64% saving in scanning, yet the loss is of 6.5%(more in Tab. 1.6.2). Current version of *ConnectEnc* application incorporates this energy efficient scan mode. We also foresee that *ConnectEnc* framework can save considerable energy when multiple P2P application are running by providing encounter information to all the application and thus preventing each of those applications from running their own scanning process.

MaxThres	Loss(W)%	Saving(W)%	Loss(B)%	Saving(B)%
3	6.52	64.21	6.79	66.31
7	10.52	75.27	11.40	76.61
15	15.11	81.53	15.02	82.29

Table 1.3: Tradeoff between saving in terms of scans and loss of information, W and B indicates Wifi and Bluetooth trace resp.

1.6.3 Simulation Evaluation

To test the utility of *ConnectEnc* recommendations on a larger scale, we use trace-based simulation. The goal of this simulation is to investigate if *ConnectEnc* recommendations can make a difference in routing messages over a Delay Tolerant Network (DTN) with selfish nodes. DTNs are infrastructure-less networks that work on the cooperation among the nodes. Since nodes spend their resources in routing messages, the nodes may only route messages for nodes they know or when they have some incentives (thus become selfish). Here we use *ConnectEnc* framework to help a node decide from which of its peers to accept packets and route it further while being selfishness to other peers. Since *ConnectEnc* selects nodes that are similar in terms of spatio-temporal similarity, several nodes having high encounter score may be already know to the user (Homophily [29]). Therefore routing messages for nodes that have high encounter score may give the user social incentive [24].

Setup: To examine the effectiveness of *ConnectEnc*, we use epidemic routing protocol [34]. Epidemic routing performs a controlled flooding and has been proved to provide lower bound in performance in terms of hops, delay and unreachability. These properties make it an appropriate tool for the purpose of our evaluations. We use WLAN traces (converted into encounter trace) from 3 campuses for this simulation.

Fig. 1.11 shows the flow chart for *ConnectEnc* routing used by each node. When a node receives a message from peer with encounter score above a thresholds (T), it accepts the packet and attempts to route it. Otherwise, the node accepts the packet based on factors such as user-configured selfishness. The selfishness is defined as the

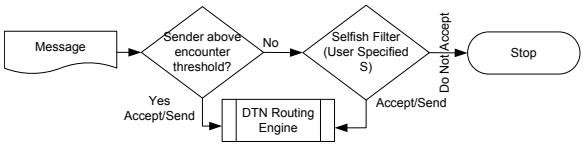


Figure 1.11: Flow chart for DTN routing using*ConnectEnc* 's peer selection

probability (S) that a node will not accept and route packets for a peer who is below a set encounter score threshold.

The performance of epidemic routing is measured using three metrics: *i.* Unreachability - the number of nodes out of all receivers that could not be reached by a given source, *ii.* Delay - the ratio of average time taken by a message to reach all the possible receivers over the max possible delay, and *iii.* Overhead - average number of hops a message took to reach all the possible receivers using the shortest path. Since overhead and delay were seen to vary directly with unreachability, we have skipped their results.

For the simulations, we use first 60 days of traces to create preliminary encounter scores and run epidemic routing on traces for next 30 days. Encounter scores are updated weekly during the run of epidemic routing (to mimic a mobile device as computing encounter scores after every encounter or daily would be resource intensive for the device). Around 800 nodes are randomly selected as sources for the epidemic routing.

Results: Intuitively, selfishness should cripple the connectivity in the network. Fig. 1.6 shows that the network unreachability increases as S increases (and $T = 0$). To the benefit of our scheme, we find that as social incentive in introduced based on the encounter scores in the network, the effect of selfishness is reduced. Here we use encounter scores from DE filter (other filters show similar trend). For U1, when $T = 0\%$ and $S = 0.9$, unreachability increases by 83% from the case when $S = 0$. However, increasing threshold to $T = 40\%$ ($S = 0.8$) unreachability remains only 31% from the case when $S = 0$. Likewise, for Dartmouth, when $T = 0$ and $S = 0.9$, unreachability increases by 40% from the case when $S = 0$. However, increasing threshold to $T = 40\%$ ($S = 0.9$) unreachability remains only 10% from the case when $S = 0$. For USC, $T = 0$ and $S = 0.9$ increases unreachability by 1.7% of the case when $S = 0$. However, increasing threshold to $T = 40\%$ ($S = 0.9$) brings unreachability to only 0.48% from the case when $S = 0$. The effect of *ConnectEnc* peer recommendations is higher when selfishness is high, which makes *ConnectEnc* more suitable in networks with high selfishness. The effect of peer selection by *ConnectEnc* (or selfishness) is not significant in USC traces, which could be a result of high unreachability in the network even at $S = 0$ (5 times of U1 or Dartmouth).

We now compare the performance of Hybrid Filters (using 5 different weight combinations). The highest unreachability (worst performance) is produced by using only the *BM* filter score and the lowest by using the FE filter (Fig. 1.12). The combination of filters at equal weights has unreachability close to FE filter. This analysis shows that, that combination of filter scores can produce better results (an also avoids user confusion) than using individual filters. Better performance of FE

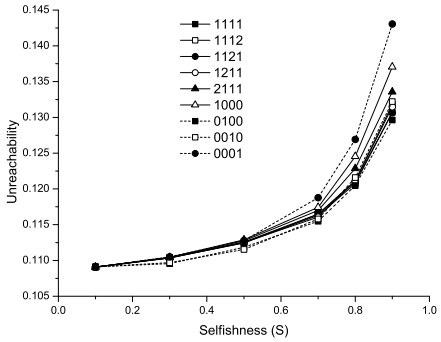


Figure 1.12: Hybrid filter results when T=40%. Number on the legend indicated the ratio of score from each filter. For e.g., 1211 implies $\alpha_{DE} = 0.2$, $\alpha_{FE} = 0.4$, $\alpha_{LV-D} = 0.2$, and $\alpha_{BM} = 0.2$ and 0100 implies $\alpha_{DE} = 0$, $\alpha_{FE} = 1$, $\alpha_{LV-D} = 0$, and $\alpha_{BM} = 0$ (Sec. 1.7.3)

over *BM* does not implies that we should not use *BM* but it implies that for this particular application FE is a better Encounter Filter.

1.7 Other Modules

This section discusses the remaining modules as mentioned in the architecture diagram (Fig. 1.1). These modules are not needed for basic functionality of *ConnectEnc*, but can enhance its capabilities. These modules include Anomaly Detection, External Inputs and Unified Score Generation. Due to unavailability of any suitable existing anomaly detection system, we have designed our own. External Inputs and Unified Score Generation are provided to give a high-level idea about the framework, however, more research in the future is needed.

1.7.1 Anomaly Detection

Incorporating resilience to attacks is a primary requirement for our design. Here, the attack on the *ConnectEnc* system includes an attempt by a peer to gain encounter score in a relatively short time by injecting many encounter events (e.g., via stalking). A growth of encounter scores in this fashion can be considered an anomaly (or an attack), and a specialized anomaly detection system is needed to combat such attacks. Since *ConnectEnc* scores individual encountered peers, at present we consider single attacker scenarios.

An attacker would want to get a high encounter score as soon as possible to have high returns for limited effort. The goal of the anomaly detection design would then be to considerably raise the level of effort needed for a successful attack, to be no less than genuine trusted nodes and friends, which may entail weeks of consistent

encounters at trusted locations by the attacker. The spatio-temporal granularity of the filters determines such attack effort and provides us with the anomaly we aim to detect. Note that in our implementation, *Unified Score Generation* takes input from the anomaly detection unit. The role of anomaly detection would then be to raise a red flag (and also lower the unified encounter score of a peer) on suspicion of attack.

Anomaly detection, theoretically, can be achieved using supervised or unsupervised learning techniques. However, due to present lack of learning data (from real attacks), we only consider unsupervised technique. Our anomaly detection investigates the evolution of encounter patterns over time (without information exchange between nodes). The anomaly detection mechanism considers the growth slope of encounter statistics (including scores generated by the *Encounter Filters*). The detection system learns normal behavior over time, and incorporates deviations from the normal to detect suspect nodes and trigger user alerts.

Based on the approach mentioned above, we have create and tested this anomaly detection system with the help of trace-driven simulations (and by creating an attacker's model). The anomaly detection, we designed, is able to detect attackers with less than 8% false positives and 6% false negatives. However, to due to lack of space, we are skipping the details.

1.7.2 External Inputs

i. Recommendation & Reputation Systems: *ConnectEnc* is designed to take inputs from existing recommendation [32, 18] and reputation systems [10]. *ConnectEnc* can also *bootstrap* a recommendation system, since recommendation system scores start to evolve only after initial direct interaction. Recommendation systems can receive peer recommendations from other peers. Reputation system can receive feedback on peers from applications and utilize it to raise overall score of a peer who has low encounter score but high reputation (or reduce the score for a peer with bad reputation).

ii. Blacklist & Whitelist: User can use these lists to explicitly add and rate (including not encountered) users. This functionality allows addition of infrequently encountered yet known peers.

1.7.3 Unified Score Generation

ConnectEnc needs to provide easily understandable information to the application or the user. Providing scores from independent modules separately may confuse the user or complicate an application design. As a first step to simplify the output, we earlier created a Hybrid Filter (HF), combining the Encounter Filter scores. A similar idea can be used to combine the scores from all the modules discussed above and generate a single encounter score for an encountered peer. The scores can be combined using the following:

$$U(P_j,\alpha,\beta,\delta) = \delta H(P_j,\alpha) + (1-\delta)(\sum_{i=1}^{m} \beta_i R_i(P_j))) \tag{1.4}$$

where $U(P_j, \alpha, \beta, \delta)$ represents the unified encounter score for an encountered peer P_j, it is always between 0 (lowest) and 1 (highest). $H(P_j)$ is the score from Hybrid Filter. β_i represent the weights for other normalized external inputs (R_i) such as anomaly detection, recommendation system, reputation systems among others. Here $\sum_{i=1}^{m} \beta_i = 1$ and $0 \leq \beta_i \leq 1$. The factor δ decides the combination ratio of Hybrid Filter and other external inputs. δ varies between 0 and 1, so the combined score is also between 0 and 1. If the peer (P_j) is included in *whitelist* then this peer automatically gets the highest encounter score. However, if a peer exists in *blacklist*, she will be always be removed before sending the list to an application or the user.

The modules discussed in this section are presented for the sake of completion and would require further research in the future (out of scope for this work). For e.g, a challenge now lies in finding out the correct weights ($\alpha, \beta, \& \delta$) to combine different inputs. These weights depend on the user and application preferences.

1.8 Conclusion and Future Work

This work introduces, *ConnectEnc*, an effective encounter based framework for making informed peer selection choices in mobile P2P applications in an efficient, privacy-preserving and resilient manner. *ConnectEnc* is driven by *Encounter Filters* that leverage increased sensing capabilities of the mobile devices and their close association with users, which enables them to capture peer similarity with encountered devices at multiple levels.

We use four novel *Encounter Filters*, based on encounter frequency, duration, location behavior-vector and behavior-matrix. The score reflects the level of similarity to aid the user or application to select peers in coordination with personal preferences, location priorities, contextual information and/or encounter based keys. The calculations are fully distributed eliminating the need for any server or trusted third party.

Three phase evaluation reveals that most filters possess high stability and form a small world among the users. A series of surveys and participatory experiments shows that statistically strong correlation exists between the filter scores and the selection of peers. This validates the Encounter Filter based approach used by *ConnectEnc*. Selfishness analysis using social incentive based epidemic routing shows that it is possible to efficiently use peer recommendations by *ConnectEnc* without sacrificing network performance in DTNs. Further, resilience to attack using anomaly detection achieves less than 10% false positives and 7% false negatives.

ConnectEnc has been designed to inspire several potential applications that can be enabled in future. However, there are a few avenues that require further research. In future, we plan to address some of these questions such as handling multiple devices belonging to a user or MAC address spoofing (several techniques exist [37]) are part of future research. Future work will include analysis of other filters for measuring behavioral similarities. We also want to develop and deploy *ConnectEnc* for popular mobile platforms and study the effect of its usage on a larger scale. There is a need to conduct more research in order to understand how to effectively leverage

P2P connections in mobile societies. We hope that this research contributes to that effort.

References

[1] CRAWDAD, August 2008.

[2] iTrust/ConnectEnc mobile app, Dec 2012. https://code.google.com/p/itrust-uf/.

[3] Bizzabo, Nov 2012.

[4] Open Garden, Nov 2012.

[5] P2P games, Nov 2012.

[6] Query by a game developer, Nov 2012.

[7] R. Albert and A. L. Barabsi. Statistical mechanics of complex networks. *Rev. Mod. Phys., Vol. 74, pp. 47-97*, 2002.

[8] Eitan Altman. Competition and cooperation between nodes in delay tolerant networks with two hop routing. In *NET-COOP*, 2009.

[9] Eitan Altman, Arzad A. Kherani, Pietro Michiardi, Refik Molva, Pietro Michiardi , and Refik Molva . Non-cooperative forwarding in ad-hoc networks. Technical report, PIMRC, 2004.

[10] Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for mobile ad-hoc networks. In *P2PEcon*, 2003.

[11] Sonja Buchegger and Jean-Yves Le Boudec. Self-Policing Mobile Ad-Hoc Networks by Reputation. *IEEE Comm. Mag.*, 43(7):101, 2005.

[12] Ronald S. Burt. Decay functions. *Social Networks*, 22(1):1 – 28, 2000.

[13] Levente Buttyan and et al. Barter-based cooperation in delay-tolerant personal wireless networks. In *WoWMoM*, 2007.

[14] Chia-Hsin Owen Chen and et al. Gangs: gather, authenticate 'n group securely. In *MobiCom '08*, 2008.

[15] Jon Crowcroft, Richard Gibbens, Frank Kelly, and Sven Östring. Modelling incentives for collaboration in mobile ad hoc networks. *Performance Evaluation*, 57, 2004.

[16] E.C.R. de Oliveira and C.V.N. de Albuquerque. Nectar: a dtn routing protocol based on neighborhood contact history. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 40–46. ACM, 2009.

[17] N. Eagle, A. Pentland, and D. Lazer. Inferring Social Network Structure using Mobile Phone Data. *PNAS*, 2007.

[18] Elizabeth Gray, Jean-Marc Seigneur, Yong Chen, and Christian Jensen. Trust propagation in small worlds. In *Trust management*, 2003.

[19] B. Han, P. Hui, VS Kumar, M.V. Marathe, G. Pei, and A. Srinivasan. Cellular traffic offloading through opportunistic communications: a case study. In *ACM Chants workshop*, 2010.

[20] Bo Han and A. Srinivasan. ediscovery: Energy efficient device discovery for mobile opportunistic communications. In *(ICNP), 2012*.

[21] Weijen Hsu, Debojyoti Dutta, and Ahmed Helmy. Profile-Cast: Behavior-aware mobile networking. In *IEEE WCNC*, 2008.

[22] Weijen Hsu and Ahmed Helmy. MobiLib, June 2008.

[23] Youngki Lee, Younghyun Ju, Chulhong Min, Seungwoo Kang, Inseok Hwang, and Junehwa Song. Comon: cooperative ambience monitoring platform with continuity and benefit awareness. In *MobiSys*, 2012.

[24] Qinghua Li, Sencun Zhu, and Guohong Cao. Routing in socially selfish delay tolerant networks. In *Infocom*, 2010.

[25] Yue-Hsun Lin and et al. Spate: small-group pki-less authenticated trust establishment. In *MobiSys*, 2009.

[26] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *LNC*, pages 239–254, 2004.

[27] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(2):72–93, 2005.

[28] Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Seeing Is Believing: using camera phones for human authentication. *Int. J. Secur. Netw.*, 4(1/2):43–56, 2009.

[29] Miller Mcpherson, Lynn S. Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[30] Sungwook Moon and Ahmed Helmy. Understanding periodicity and regularity of nodal encounters in mobile networks: A spectral analysis. In *GLOBECOM 2010*.

[31] Vikram Srinivasan Pavan, Vikram Srinivasan, Pavan Nuggehalli, Carla F. Chiasserini, and Ramesh R. Rao. Cooperation in wireless ad hoc networks. In *IEEE Infocom*, 2003.

[32] Glenn Shafer. Perspectives on the theory and practice of belief functions. *Int. Journal of Approximate Reasoning*, 1990.

[33] G.S. Thakur, M. Sharma, and A. Helmy. Shield: Social sensing and help in emergency using mobile devices. In *GLOBECOM*, pages 1–5, 2010.

[34] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke University, 2000.

[35] Wei Wang, Vikram Srinivasan, and Mehul Motani. Adaptive contact probing mechanisms for delay tolerant applications. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom, 2007.

[36] Jie Wu and Yunsheng Wang. Social feature-based multi-path routing in delay tolerant networks. In *INFOCOM, 2012*.

[37] Kai Zeng, Kannan Govindan, and Prasant Mohapatra. Non-cryptographic authentication and identification in wireless networks. *Wireless Communications*, 2010.

[38] Sheng Zhong, Jiang Chen, and Richard Yang. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. In *INFOCOM*, 2002.