The Vicinity Package for Opportunistic Networks

Tiphaine Phe-Neau

The Vicinity Package for Opportunistic Networks

Tiphaine Phe-Neau
UPMC Sorbonne Universités
tiphaine.phe-neau@lip6.fr

***Abstract*—Opportunistic networks have grown into a major trend in the wireless network research field. In our work, we relaxed the traditional definition of contact and intercontact times by bringing the notion of *vicinity* into the game. We first propose to analyze opportunistic/disruption-tolerant networks (DTN) using a node-centered vicinity point of view i.e. we consider that nodes are in κ-*contact* when they remain within a few hops from each other and in κ-*intercontact* otherwise (where κ is the maximum number of hops characterizing the vicinity). Second, we focused on the inner vicinity behaviors namely Vicinity Motion and on how we could reproduce these patterns to generate dataset-based synthetic pairwise vicinity movements. Finally, since the main use of opportunistic networks is to transmit data, we investigated the use of vicinity knowledge into pairwise shortest distance prediction and found out a very interesting heuristic relying on Vicinity Motion. We packaged most of our contributions concerning vicinity knowledge for opportunistic networks in the Vicinity Package written using Python 2.7 and libraries such as NetworkX and NumPy. This paper reviews the theoretical knowledge required to use the Vicinity Package.**

I. Introduction

In our modern society, citizens tend to have more and more connected devices. Through these devices, they require to always be connected to the current trends or news and they want to be able to communicate with other persons whether they are commuting, at work, or even on holidays. This need of a "super-connectivity" shows its limits with the resulting telecommunication companies infrastructure overload [1]. On the lookout for alternative means to provide data to users, we find an attractive solution with the DTN paradigm. It is mostly user-based, does not need an overall infrastructure, takes advantage of user mobility as a transmission catalyst, and manages to deliver an interesting amount of information in the network.

This all started high in the sky with satellite networks and the idea of an Interplanetary Internet [2]. To work with such type of networks is clearly different from working with our common wired or Wi-Fi networks. Compared to most systems, interplanetary networks do bear unusual features. For instance, topological distance between spatial nodes is often of thousands of kilometers inducing long delays between the emission and reception of a signal. These long delays are considered faulty in usual networks. However in this case, it is a natural part of their functioning. The opportunistic networking research area gained attention in 2003 when Kevin Fall formalized the characteristics of a "delay-tolerant network architecture" for challenged networks [3]. By bringing a back-to-earth vision to challenged networks, Fall sparked a lot of interests in our fellow scientists. By lifting a few technical constraints, our community could extend wireless networks use to such challenged networks. This new paradigm could be applicable to urban areas with urban nomads always carrying connected devices (laptops, smartphones, etc.) When two or more of these devices are close enough, they have potential connectivity and transmitting powers thanks to their embedded technologies. Transmitting information hop-by-hop between these moving devices becomes theoretically possible. Such challenged networks are called "disruption-tolerant networks" (DTN) or "opportunistic networks".[1]

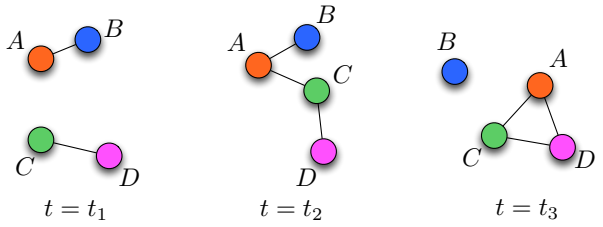


Fig. 1. An example of disruption-tolerant/opportunistic network.

Opportunistic networks rely on device's "short" range connectivity (currently Bluetooth, NFC or Wi-Fi Direct) to transmit data. Therefore, nodes can transmit data only when they are close enough i.e., in *contact*. Fig. 1 presents an example of a 4-node DTN. At time t_1, A and B as well as C and D are connected. Next at t_2, they form a chain and finally at t_3, B moves away leaving A, C, and D fully connected. Any disruption-tolerant protocol should cope with such a connected/disconnected scenario. To reach the destination, nodes use a hop-by-hop "store, carry, and forward" scheme. DTN have different characteristics from other networks like Wi-Fi, 2G, 3G or wired networks, therefore, their characteristics need to be thoroughly understood before we are able to use them. Many studies have shown the clear potential of DTN as a self-dependent network model. In our work, we proposed the utilization of the vicinity in DTN to improve its characterization, understanding, and functioning.

In this paper, we summarize the knowledge required to understand and use the Vicinity Package for opportunistic networks. First, we define the notion of vicinity in opportunistic networks and examine some of its properties. Then, we focus on the inner vicinity dynamics and model it via a chain model namely the Vicinity Motion. Next, we analyze the prediction capacities embedded in the Vicinity Motion model and show

[1]We will alternatively be using these two terms in this paper.

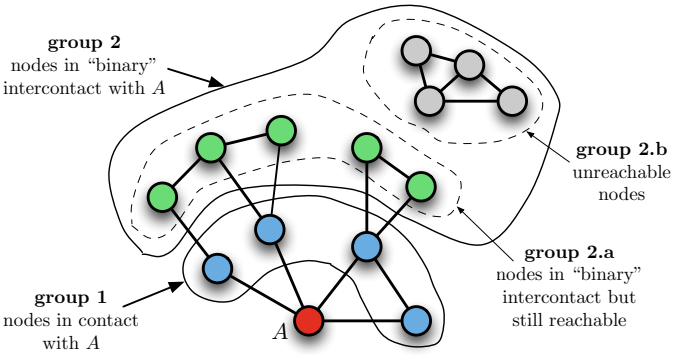


Fig. 2. Motivating example. From node A's point of view, we see that nodes in *group 1* are in contact. Using the usual binary vision, we conclude that all other nodes are in intercontact (i.e., in *group 2*). However, nodes in *group 2.a* are essentially different from nodes in *group 2.b*. A has end-to-end paths toward the first and no paths at all to the latter.

its efficiency. For most of these analyses, we have implemented the corresponding algorithms in the Vicinity Package. Finally, we give an overview about our implementation choices and the different various algorithms. For a more detailed version on vicinity knowledge in opportunistic network, please refer to [4].

II. Vicinity: Definition & Timelines [5], [6], [7]

As a descendent of historical networks where the notion of contacts and direct connectivity is prominent, the first characterization of opportunistic networks focused on contacts between nodes and their resulting intercontact periods. The notion of contact has been well investigated years before [8], [9], [10]. However, the intercontact notion is quite unexplored. The first sensible approach was to consider intercontact as the complementary of contact. This assumption was maintained in the context of a number of interesting studies but it may be too shallow to correctly reflect the underlying network topology.

In Fig. 2, we represent a network snapshot illustrating our concerns. This figure represents a network where nodes in *group 1* are in contact with A (i.e., they are within A's direct communication range). In the usual *binary vision*, all remaining nodes (*group 2*) are, by definition, in intercontact. Still, we notice that there is a fundamental difference among nodes in *group 2*. None of the nodes in *group 2.a* are in contact with A; nevertheless, they do have a contemporaneous path to A. On the other hand, nodes in *group 2.b* do not have any path to A. In opportunistic networking where we need to gather as much knowledge as possible to achieve efficient communication standards, deeming both cases of intercontact under the same definition results in a waste of information. Suppose A needs to send a message to one of the nodes in *group 2.a*. In such a situation, most DTN approaches infer the impossibility of exchanging messages via multi-hop paths and often calls for a "wait" period until it meets the destination or finds someone else that knows the destination "better". With this binary vision, A does not know that the destination is nearby, and may miss an opportunity to communicate if, for example, the destination moves after some time to *group 2.b*. Noticing that contemporaneous paths may exist between nodes is important. Neglecting such closeby possibilities is a waste of connectivity assets in DTN.

A. Datasets

For our study, we observe the binary assertion and vicinity properties in real-world experiments as well as a synthetic datasets. We use realistic measurements to observe the extent of vicinities in real-life situations. We also confronted the vicinity notion to a synthetic datasets to observe its presence in a dedicated mobility pattern.

Infocom05 measurement was held during a 5 day conference in 2005 [11]. 41 attendees carried iMotes collecting information about other iMotes nearby within a 10m wireless range. We study a 12-hour interval bearing the highest networking activity. Each iMote probes its environment every 120 seconds. *Infocom05* represents a professional meeting framework.

Sigcomm09 counted 76 attendees with dedicated smartphones probing their surroundings during 5 days [12]. Smartphones sensed their surroundings using Bluetooth every 120 seconds. *Sigcomm09* is another example of a professional meeting scene.

Rollernet had 62 participants measuring their mutual connectivity with iMotes during a 3 hour rollerblading tour in Paris [13]. These iMotes sent beacons every 30 seconds. This experiment shows a specific sport gathering scenario.

Shopping used 25 dedicated devices in a shopping mall over 6 days [14]. Galati and Greenhalgh gave 25 devices to shop owners and planted 8 others at various locations in the mall. Devices performed neighborhood discovery every 120 seconds. *Shopping* reflects the working day routine of shop owners as well as some of their customers.

Unimi is a dataset captured by students, faculty members, and staff from the University of Milano in 2008 [15]. The experiment involved 48 persons with special devices probing their neighborhood every second. *Unimi* provides a scholar and working environment scenario.

RT (S) for *RandomTrip* is a mobility model correcting flaws from the Random Waypoint model [16]. We sampled the behavior of 20 nodes following this model on a surface of 50 $\times$ 60 m^2 with speed between 0 and 7 m/s and a 10m wireless range. We choose to simulate 20 nodes over this surface to recreate office conditions.

B. The Binary Assertion Issue

Considering the notion of intercontact as the mere binary complementary vision of contact is understandable. The leading property of historical networks has always been the "contact" between nodes. But in challenged networks such as DTNs, we have to get the most of every situation and surrounding assets. In Fig. 2, we observed that there were unused pairwise connectivity between nodes. The traditional contact vision misuses end-to-end "not-in-contact" connectivity. The *binary assertion issue*, where we ignore end-to-end connectivity beyond one hop, brings an interesting

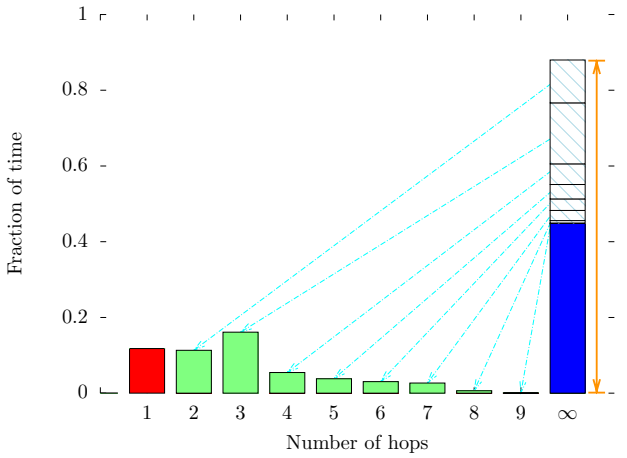


Fig. 3. Example of time-distance distribution from the *RT* dataset. On the left, we see that nodes spend 10% of their time in contact (1-hop). With the binary vision, we then consider that nodes spend around 90% of their time in intercontact (the double arrow on the right). With vicinity goggles we see that in reality, they dwell at a distance 2 for around 10%, at a distance 3 for 16%. Real intercontact deprived of multi-hop path represents only 50% of the time (∞).

interrogation: *how pervasive are these hidden communication possibilities?*

To understand the problem, let us show an example for a given pair of nodes using the *RT* dataset. We compare the cumulated amount of time they spend in contact and in intercontact and plot the results in Fig. 3. We observe that nodes spend around 10% of their time in contact and around 90% in binary intercontact. If we consider the vicinity aware vision, for the same pair of node, we realize that they spend around 10% of their time at a 2-hop distance, 18% at 3-hops, 5% at a 4-hop distance, etc. The true time they spend without any path to one another is only around 50% of the experiment duration. The binary assertion hides 40% of the time where these two nodes have a path connecting them.

More than just limiting our vision, the binary assertion prevents us from leveraging our environments and performing simple yet efficient closeby end-to-end transmissions. To be able to use and characterize the closeby topological paths, we first define the notion of vicinity in DTN.

C. The Notion of Vicinity in opportunistic networks

To the best of our knowledge, this is the first time the notion of vicinity has ever been formalized in DTN. To understand the extended transmission possibilities in opportunistic networks, the first issue is to provide a formal definition of what the notion of "nearness" means in DTNs. The κ-vicinity notion brings an ego definition to DTNs and also adds a hop-based discrimination [5], [17], [6]. This differentiation helps us limit our vision according to our needs as well as identifying neighbor properties. We discriminate a node i's vicinity according to the number of hops between i and its surrounding neighbors. We use the instantaneous connectivity graph between nodes to compute pairwise shortest paths. This connectivity graph illustrates the current network state and what is immediately useable.

Definition 1: **κ-vicinity.** The κ-vicinity $\mathcal{V}_\kappa^i$ of node i is the set of all nodes with shortest paths of length at most κ hops from i.

Clearly, $\mathcal{V}_{\kappa-1}^i \subset \mathcal{V}_\kappa^i$. In Fig. 4, we illustrate the node i's 1-vicinity and 2-vicinity at instant t. This is an interesting point of view for opportunistic networks because it extends a node's knowledge to immediately useable communication opportunities. The κ-vicinity empowers a node's reach in the network [7].

Vicinity knowledge may come from different techniques. For instance, we can use link state protocols to gather information about a node's connected component. There are many ways to do so, but they all are costlier than getting information from contacts only. The tradeoff between getting vicinity information and its additional costs may be a reason not to use κ-vicinity. However, we provide a solution to this tradeoff by suggesting that monitoring the $\{3, 4\}$-vicinity is enough to get most events in a node's surroundings for the datasets we consider in our analyses [5], [7].

The κ-vicinity defines a node's neighborhood, i.e. its new *zone* in the network. To characterize this zone's relationships to node i like the *contact* and *intercontact* notions previously, we must define some temporal measures relating to time neighbors spend in the zone and time outside the zone, namely "κ-contact" and "κ-intercontact". We maintain a pairwise definition for these measures and assume that connectivity is bidirectional.

Definition 2: **κ-contact.** Two nodes are in κ-contact when they dwell within each other's κ-vicinity, with $\kappa \in \mathbb{N}^*$. More formally, two nodes i and j are in κ-contact when $\{i \in \mathcal{V}_\kappa^j\} = \{j \in \mathcal{V}_\kappa^i\}$. In other words, a contemporaneous path of length at most κ hops i and j.

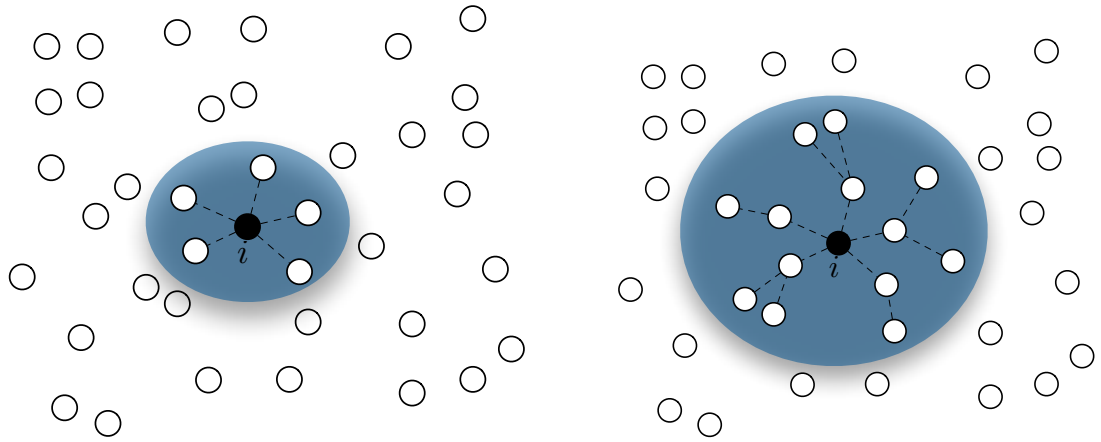
We also need to grasp the intercontact observations for our vicinity viewpoint. The literature definition of mere intercontact is when two nodes are not in contact. Therefore, we consider κ-intercontact when two nodes are not in κ-contact. These are complementary notions.

Definition 3: **κ-intercontact.** Two nodes are in κ-intercontact while they do not belong to each other's κ-vicinity. Formally speaking, two nodes i and j are in κ-intercontact when $\{i \notin \mathcal{V}_\kappa^j\}$ or $\{j \notin \mathcal{V}_\kappa^i\}$ or there is no path of length κ or less linking i and j.

Note that 1-contact matches the contact notion and 1-intercontact corresponds to usual binary intercontact.

D. S1: Vicinity pairwise timelines

To understand the properties of vicinities in opportunistic networks, we rely on a key data structure called "timeline". This timeline will be the first output of the Vicinity Package. It takes contact traces as inputs. We first organize the trace as a chronological sequence of instantaneous events. Events can either be a link appearing or vanishing between a pair of nodes (i, j) at time t. We symbolize this type of event as



(a) 1-vicinity (b) 2-vicinity

Fig. 4. κ-vicinity illustration. Node i's $\{1, 2\}$-vicinity at a given time t.

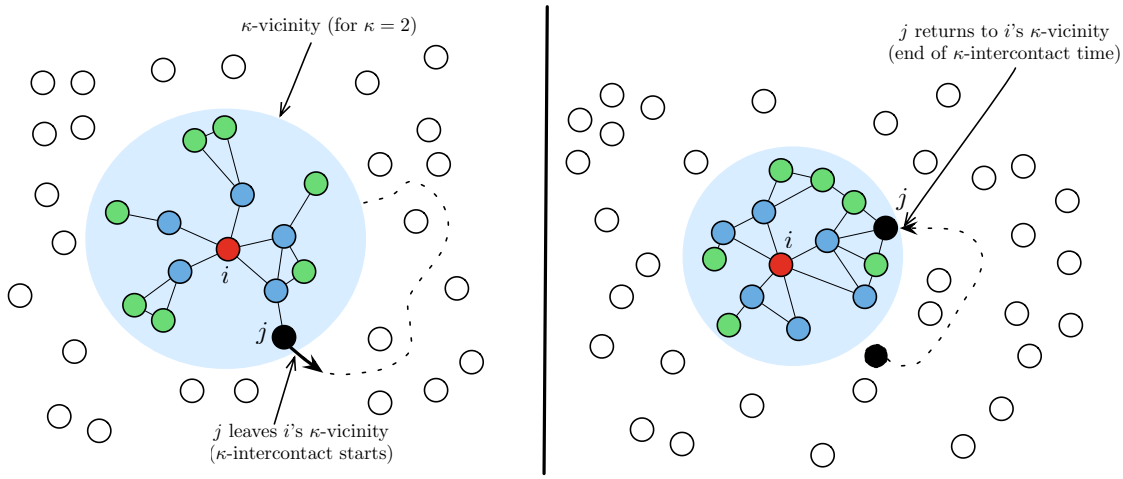


Fig. 5. Node i's κ-vicinity and the κ-intercontact phenomenon. For the sake of clarity, we only display i's connectivity links within the κ-vicinity.

$e =< t, i, j, \texttt{UP/DOWN} >$. `UP` indicates the appearance of a link between i and j and `DOWN` its disappearance.

For a given pair of nodes (i, j), a timeline consists in the sequence of their mutual shortest distance through to time (see Fig. 6). Formally speaking, we represent timelines as a sequence of tuples $< n, i, j, t_{begin}, t_{end} >$. This means that between t_{begin} and t_{end}, nodes i and j are at a n-hop distance.

All timelines are initialized with a starting tuple $< \infty, 0 >$ indicating that they are in κ-intercontact '∞' at time 0. Following tuples indicate a change in this state and the time at which it occurs. As long as there are events in the trace, we read them and update the adjacency matrix before computing pairwise shortest distances. Then, we update the corresponding pairwise timelines accordingly. Finally, we format and print gathered data into timelines.

E. Examples of use

Missed transmission possibilities. To quantify how many end-to-end transmission opportunities the binary assertion misses, we present what we call aggregated network sociostructures in Fig. 7 [6]. For *Infocom05*, we plotted (in layered mode) the number of connected pairs for each shortest distance. The bottom layer symbolizes the amount of pair of nodes in contact. Layer 2 shows the amount of pairs connected via a 2-hop path, layer 3 represents connection via a 3-hop path, and so on. Each sociostructure layer of value ≥ 2 represents pair of nodes linked by end-to-end paths longer than 1 hop. Recall that the binary assertion does not recognize such relations.

In Fig. 7, for *Infocom05*, we observe several density peaks of connected pairs. Being a conference-based measurement, these peaks indicate morning arrivals, lunch, the afternoon break, and end of sessions. During high density peaks, an unexpected observation is how pairs connected by 2 hops overcome contact opportunities. Places with high density ignite transmission possibilities beyond mere contact. As a result, in such a scenario, *2^+-hop transmissions should be more helpful than mere contact transmissions* or pure DTN techniques.

The presented sociostructures illustrate the illusion provided by the binary assertion. If we maintain contact-only knowledge

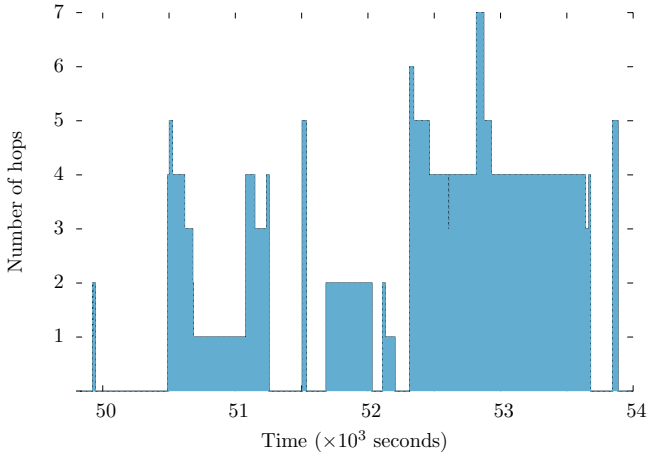


Fig. 6. A pairwise timeline from the *Unimi* dataset. From 50,000 seconds to 50,500 seconds, the two nodes did not have a path to one another. Then, they briefly were at a 5-hop distance before coming closer at a 4-hop and then a 3-hop distance and so on.

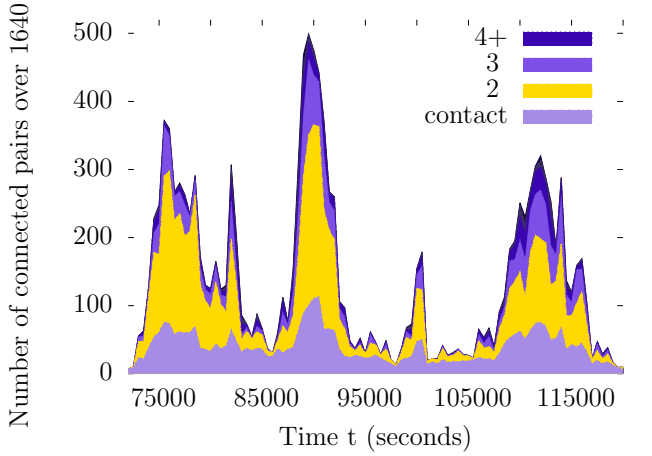


Fig. 7. *Infocom05* sociostructures presenting the amount of pairs connected by contacts, 2-hop paths, 3-hop and so on in a layered mode according to time. We notice the omnipresence of pairs connected by 2^+ hops. They often overcome the possibilities offered by contact only (bottom layer). As a result, contact opportunities only represent a minor part of all end-to-end opportunities between two nodes. The binary assertion overlooks these possibilities by blending all nodes in intercontact under a unique concept.

in DTN, we miss the omnipresent power of nodes at 2^+ hops. These 2^+ layers represent powerful transmission opportunities as they only involve few relays that could reduce significantly end-to-end delays. These layers vary in importance but are almost always present. Considering only contacts provides a minor vision of what happens in the network. Observing a node's vicinity at a 2-hop distance may more than double the transmission opportunities as seen in the *Infocom05* dataset at 90,000 seconds. The binary assertion weakness highlights the importance of observing nodes beyond simple contact. In social settings, there may be a concentration of people around us (when commuting or at work), yet we limit our vision to contacts only while there is so much more at hand.

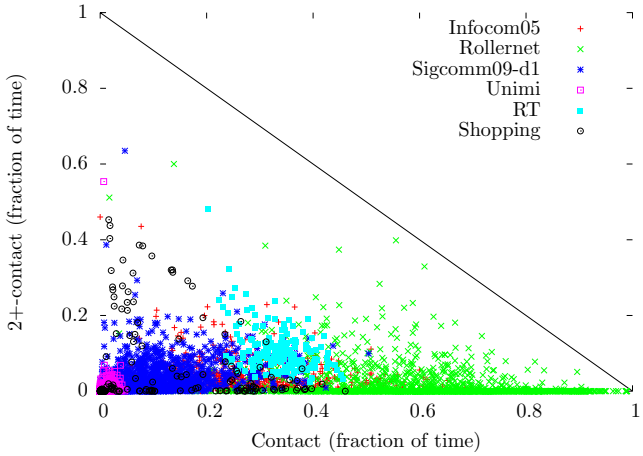


Fig. 8. Pairwise behavior according to the fraction of contacts and 2^+-contacts. Each dot represents a pair of nodes. On the x-axis, we have the fraction of time they dwell in contact. On the y-axis, the fraction of time they observe κ-contacts. Note that, for the same density of contact, we can obtain a wide difference in 2^+-contact percentage.

Pairwise behavior variability. In Fig. 8, for all pairs of nodes, we plot on the x-axis the fraction of time they spend in contact and on the y-axis the fraction of time they spend in 2^+-contact. We visualize a wide variety of meeting patterns with many pairs having long 2^+-contacts.

Using timelines, we observe that a large portion of nodes display a significant fraction of time with end-to-end transmission capacities endorsed by contact and κ-contact. Nodes bearing end-to-end transmissions beyond contact for more than 10 minutes are as follows: 78.3% for *Infocom05*, 99.4% for *Rollernet*, 57.1% for *Sigcomm09*, 57.0% for *Unimi*, 100% for *RT*, and 73.7% for *Shopping*. If we increase the threshold to 20 minutes, the proportion of pair of nodes with extended end-to-end properties do not really change for the following datasets: 78.3% for *Infocom05*, 98.2% for *Rollernet*, 53.4% for *Unimi*, and 100% for *RT*. The values for *Sigcomm09* and *Shopping* decrease to respectively 44.8% and 57.7% but still remain quite high.

III. Inner Vicinity Dynamics [18]

To understand the movements happening in vicinities, we developed a model capturing inner κ-vicinity movements called Asynchronous Vicinity Motion (AVM). The asynchronous vicinity motion is a chain model using pairwise distance values as state values. Those states are linked to each other via transitional probabilities. Nodes can move sequentially from one distance to another. Our main observations are: the existence of three main movements patterns called *birth*, *death* and *sequential* movements. We show how sometimes by considering only death and sequential movements, we have more than 80% of vicinity movements. We also show how most κ-vicinity arrivals aka births occur at distance 3 or 4 and not in contact for the considered datasets [7], [5]. Based on this observations and asynchronous vicinity motion transitional probabilities, we propose TiGeR, a pairwise vicinity behavior

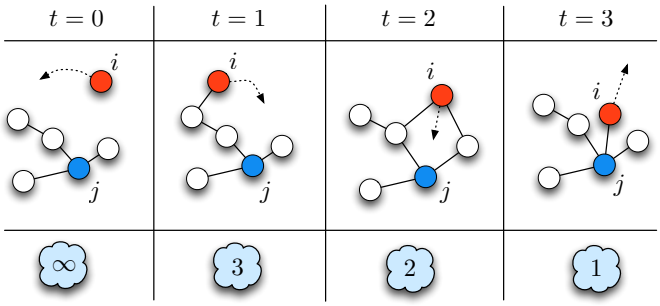


Fig. 9. An example of asynchronous vicinity motion knowledge. At $t=0$, node j is outside i's vicinity but coming closer. At $t=1$, j pops into i's vicinity at a 3-hop distance. At $t=2$, j moved closer to i at a 2-hop distance and even arrives in contact at $t=3$.

generator who generates model pairwise timelines. We can use these timelines to understand synthetic κ-vicinity functioning and test opportunistic protocols relying on κ-contacts.

A. Why Vicinity Dynamics?

As shown in Section II-B, nodes in direct contact represent only a small part of all opportunistic communications in DTN. To leverage such unused connectivity, we propose to understand how neighbors move within a node's vicinity. In Fig. 9, we illustrate the evolution of a small network. At $t=0$, nodes i and j have no path to each another – they are in intercontact. At time $t=1$, nodes i and j are not in contact (1-hop distance) but are linked via a 3-hop path. At $t=2$, i and j are at a 2-hop path and they finally come in contact at $t=3$. The usual contact/intercontact vision would consider the time steps $t=\{0,1,2\}$ as the same, i.e., that i and j are in intercontact. Instead, when using the vicinity notion, such an "extended" view of communication opportunities is taken into account. Opportunistic networks can benefit from contacts that were not used before.

B. The Asynchronous Vicinity Motion Framework

The asynchronous vicinity motion (AVM) framework analyses vicinities (the κ-vicinity) for a given network. We want to answer the following question: *when the distance n between nodes i and j change, what is the probability that their distance becomes m, with $m \neq n$?*

In the remaining of our work, n is both the mutual shortest distance for a pair of nodes and the vicinity chain state while κ is the max hop distance in a κ-vicinity ($n \in (\{1,...,\kappa\} \cup \{\infty\})$). We name the period between two changes in a pair's shortest distance a *step*. Note that AVM's step duration depends on the network dynamics, therefore, steps do not have a constant duration. AVM analyzes all changes in an asynchronous way. To answer the aforementioned question, we follow a two-stage methodology:

1) **Timeline generation.** Details in Section II-D.
2) **Vicinity analysis.** Timelines provide the necessary information to characterize the transition probabilities between given distances.

We recapitulate the whole workflow in Fig. 10.

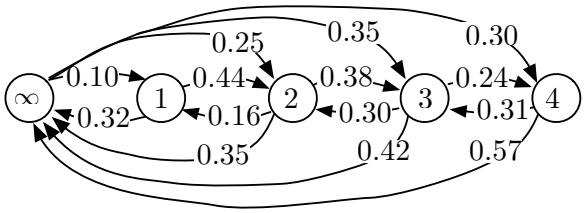


Fig. 11. Infocom05 average asynchronous vicinity motion for a pair (i,j) and $\kappa=4$. For the sake of clarity, we display only a few transitions. The probability of a node appearing in contact $\{\infty \rightarrow 1\}$ is 10% or when nodes are at a 3-hop distance, the probability for them to be next at distance 2 is 30%.

Vicinity analysis. To illustrate AVM, we use a chain process for *each* pair of nodes. For a given node i, let $X^s_{i,j}$ describe the distance between nodes i and j at step s. The vicinity analysis step (2) takes timelines as input and provides the corresponding transitional probabilities for vicinity chains. We describe the two main component type of our chain process here:

- **States.** The chain states depends on the κ we choose, i.e., the size of the vicinity we wish to monitor. The number of states is $\kappa+1$; the first state, denoted '∞', corresponds to the case where the two nodes are in κ-intercontact. The state $\{1\}$ represents a contact and the remaining states $\{2,\ldots,\kappa\}$ correspond to a situation of κ-contact where the exact distance between nodes is the corresponding state.
- **Transitional probabilities.** To understand AVM, we concentrate on the chain conditional probabilities between states, i.e., the probability of two nodes being at a distance of m at step s knowing that they were at a distance n in the previous step $s-1$: $\mathbb{P}(X^s_{i,j}=m \mid X^{s-1}_{i,j}=n)$, $m \neq n$.

As an example, we show in Fig. 11 the average transitional probabilities of AVM for *Infocom05*. For the sake of clarity, we omit certain transitions. As we can see, when nodes i and j are in κ-intercontact '∞', the probability that they meet directly is 10% while it is 35% for a 3-hop distance. This appearance behavior varies from one dataset to another and highlights the utility of the Vicinity Package to easily gather data.

C. Example of use

Vicinity patterns. In the analyzed datasets, we observe three main types of transitions, namely *birth, death*, and *sequential movements*.

1) Birth in the κ-vicinity: Birth is the phenomenon of appearance in the κ-vicinity after a period of κ-intercontact. The main interest of such knowledge is for a node or a protocol to know at which distance another node may appear. Imagine in the *Infocom05* dataset that node i wants to send a message to node j, which is currently outside i's κ-vicinity. We now know that j will appear with a probability of 20% at a 3-hop distance (see Fig. 11).

In Fig 12(a), we present the values concerning the birth motion for our datasets. On the x-axis, we represented the actual incoming state (the distance at which a node appears). On the

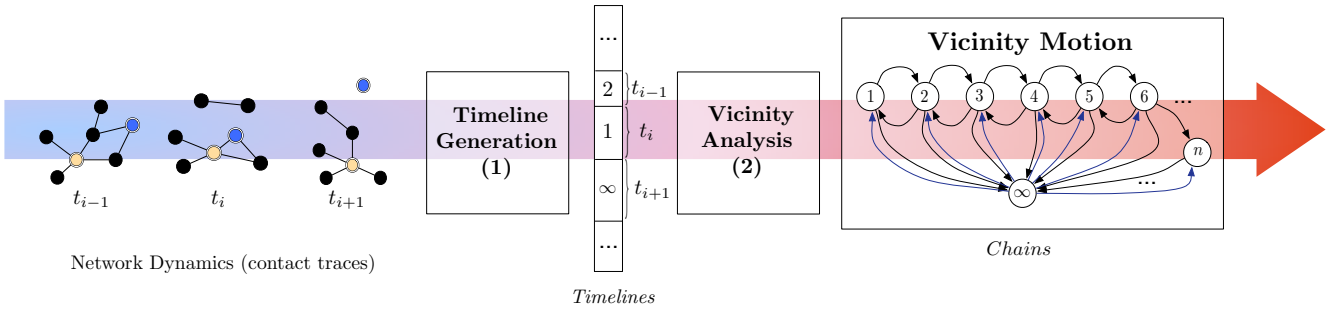
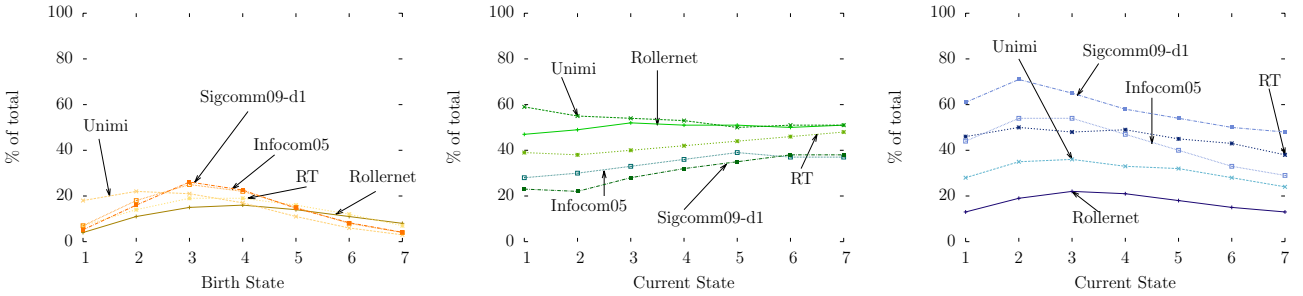


Fig. 10. Vicinity motion generation workflow. We begin by reading Network Dynamics under the form of contact traces describing network connectivity through time. We process them using (1) the timeline generation module. This stage produces *timelines*. Step (2) aka Vicinity Analysis examines these sequences to compute transitional probabilities and corresponding vicinity motion *chains*.



(a) Proportion of birth values.

(b) Proportion of deaths with regard to the full chain.

(c) Proportion of sequential movements with regard to the full chain.

Fig. 12. Birth, death, and sequential rates.

y-axis, we present the actual birth transitional probability for each distance. For all datasets, the highest birth probability belongs to the set $\{1, 2, 3, 4\}$. The cumulated transitional probabilities up to 4 represent from 50% to 70% depending on the dataset. For a random dataset, if we had chosen to extend these probing limits only to a state 4, we would detect from 50% to 70% of nodes vicinity appearance.

2) Death in the κ-vicinity: Death is when nodes vanish from the κ-vicinity. Being able to foresee death movements, i.e., a node being in κ-intercontact can indicate when to begin a fully opportunistic routing technique. As long as nodes are in the vicinity, we can use end-to-end paths towards them. However, when we suspect that nodes will next be out of the κ-vicinity, it may be time to trigger a different routing approach. Birth and death events represent a big share of the movements alongside sequential movements as presented next.

3) Sequential movements: For two nodes, sequential movements consist in the process of drifting closer or further from each other using adjacent states of the chain: when nodes (i, j) are at a 4-hop distance, they sequentially move closer if they are at a 3-hop distance during their next step, they sequentially drift away if they are next at a 5-hop distance.

A non-negligible part of vicinity movements stems from sequential behaviors (see Fig. 12(c)). For *Unimi* and *Infocom05*, as long as nodes stay in the κ-vicinity, sequential movements represent between 50% and 80% of movements. We call erratic or random movements, all movements that are not birth nor death nor sequential. They represent a minor share of AVM and can be overlooked as predicting their destination is tougher and brings only marginal knowledge gains.

In Fig. 13, we display the proportion of death, sequential, and erratic movements (from bottom to top) among all vicinity moves for *Infocom05* which is representative of other datasets. Erratic movements grow with the distance between the nodes, while death processes remain stationary around 30%. Sequential movements are strong within the 4-vicinity. The further two nodes are, the higher the proportion of erratic movements. Wider vicinity bears fickler connectivity at the borders and more random hopping.

D. S2: TiGeR– Synthetic Timeline Generator

The direct application of AVM analyses is the possibility of generating synthetic timelines. Timelines embody the pairwise vicinity behavior. TiGeR (TImeline GEneratoR) relies on the asynchronous vicinity motion module outputs (extracted timelines and transitional probabilities). The use of timelines to bootstrap nodes' vicinity knowledge into opportunistic networks is original. Before, protocols like BUBBLE Rap used history of nodes contact periods in order to predict future encounters [10]. Now, instead of focusing on contacts only, we extend this knowledge to the node's κ-vicinity. Vicinity provides more network knowledge and therefore multiplying the possibilities of encountering another node [7]. To generate pairwise vicinity behavior, TiGeR relies on transitional probabilities, a given κ value and κ-contact durations distributions.

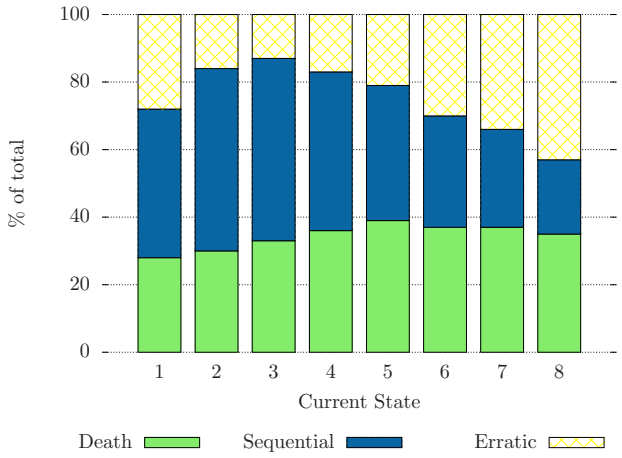


Fig. 13. Repartition of births, deaths, and sequential movements in *Infocom05*.

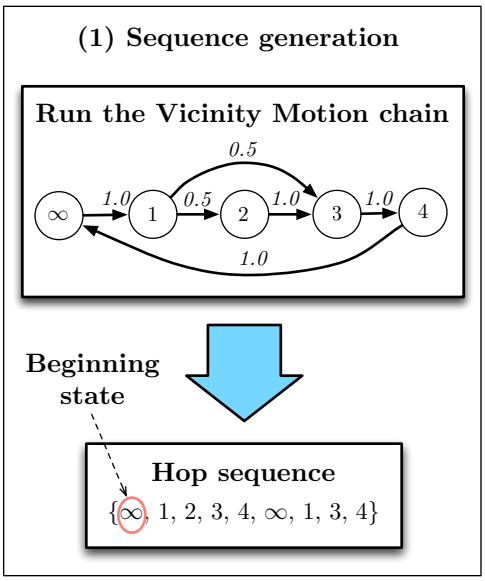


Fig. 14. TiGeR's hop sequence generation example. From the given asynchronous vicinity motion transitional probabilities, TiGeR produces a possible hop sequence s. s has transitional probabilities similar to the initial asynchronous vicinity motion transitional probabilities.

Based on AVM's transitional probabilities, we generate a sequence of pairwise shortest distance. The challenge is then to match these timelines distances into correct interval durations as well as plausible distance intervals number.

1) Hop sequence generation: This step generates an AVM transition compliant hop sequence (a list of distances whose AVM transition will be similar to the provided transitions). We take a max distance D and process the provided AVM transitions as follows:

- **Beginning state.** We need to bootstrap the generated timeline with a first starting distance. We choose to get a random starting distance denoted d_0 among all the existing states $\{\infty, 1, ..., D\}$. For example, let us begin with d_0 = '∞'.
- **Run the AVM chain.** We run the corresponding AVM chain from the starting state d_0 = '∞'. We choose the highest outgoing probability from ∞ and decrement the taken transitional rates by a certain value Δ. In TiGeR, we set Δ to be the greatest common factor among all transitional rates. When we find ourselves to be in a sink node (all output transitional rates are null), we randomly choose another output state. We stop the distance generation when all the transitional rates are depleted.

We then repeat the processes for all the max distance values in [1:D]. Considering the max distance distribution, we can generate several synthetic timelines. The only precaution to take is to normalize the corresponding AVM transitional probabilities before running the chain.

We detail an example from Fig. 14. For a max-min distance equals to 4, me assume the following transitional rates: $\{$ ($\infty \rightarrow 1 = 1.0$), ($1 \rightarrow 2 = 1.0$), ($2 \rightarrow 3 = 0.5$), ($2 \rightarrow 4 = 0.5$), ($3 \rightarrow 4 = 1.0$), ($4 \rightarrow \infty = 1.0$) $\}$ all other transitional probabilities are considered null here. We start with d_0 = '∞'. We determine Δ = 0.5 (because it is the highest common factor among $\{1.0, 0.5\}$). From the AVM in Fig. 14, we take the transition '$\infty \rightarrow 1$'. The resulting AVM is the same as before except that the '$\infty \rightarrow 1$' transition value is now 1.0-0.5 = 0.5. We normalize this value by the total outgoing probabilities and '$\infty \rightarrow 1$' becomes 1.0. We are now in state 1 and can decide to go either to state 2 or 3 because they have the same outgoing probability 0.5. We randomly choose state 2 and decrease the '$1 \rightarrow 2$' to 0.0 and normalizing '$1 \rightarrow 3$' to 1.0. Then from state 2 we got to 3 and so on, until all transitional probabilities are ≤ 0.0. In our case, the matching hop sequence s would be s =$\{\infty$, 1, 2, 3, 4, ∞, 1, 3, 4$\}$. Now that we have s, we need to match these states/distances sequence with accurate intervals durations.

2) Time matching: Using hop sequence s, we match each of its distances with a plausible interval duration. Depending on the user need, TiGeR provides two modes. First Mode (I) mimics timelines with life-like interval durations while the second Mode (II) outputs timelines with more AVM compliant transitions. This step requires the user to give $\mathcal{L}$ the timeline length he wants to get. We call the timelines generated with Mode I, MI-timelines, and those by Mode II, MII-timelines.

- **Mode I reflects plausible intervals duration.** The first available option means to reflect the κ-interval durations. For each distance from s, we use the κ-contact duration distributions from the AVM module. Let us say that $s = n$, then we use a Gaussian distribution based on the n-contact duration distribution (average duration, first and third quartile) to extract a plausible interval value. Then, we record and sum the obtained durations until the total intervals duration exceeds $\mathcal{L}$. MI-timelines may lack some step from s but they respect the required duration $\mathcal{L}$ and plausible interval durations.
- **Mode II focuses on transitional probabilities.** In the second option, we focus on respecting AVM transitional

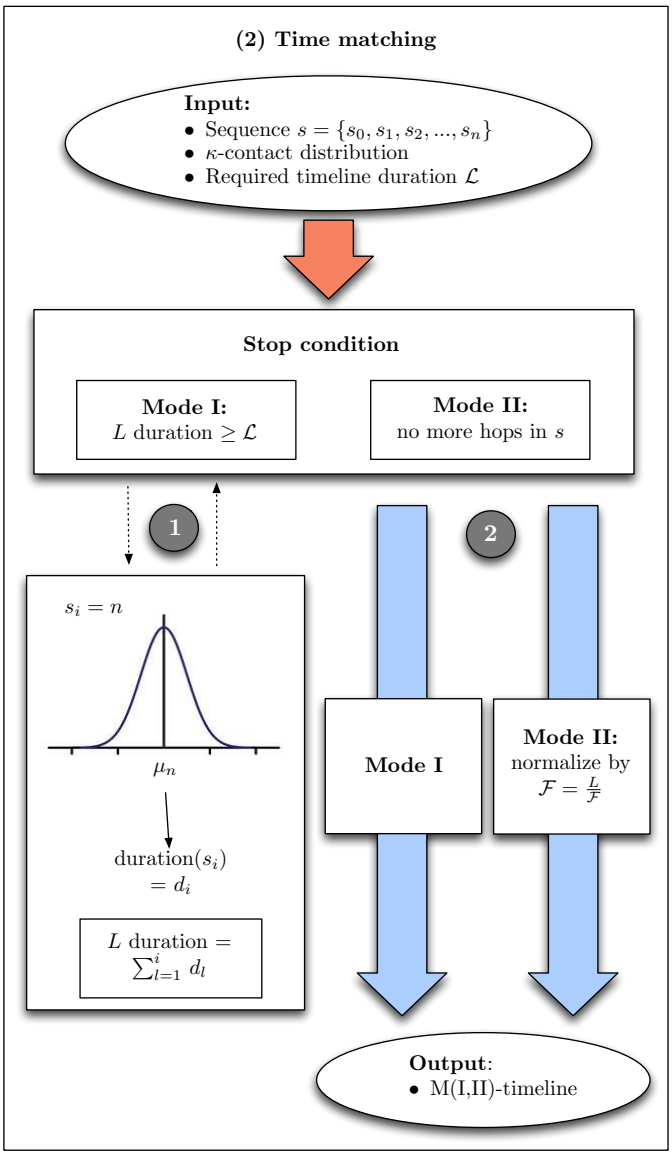


Fig. 15. TiGeR's time matching process.

probabilities. We keep the same process as in Mode I without limiting the time matching to the $\mathcal{L}$ duration. We keep on generating the κ-intervals durations to plausible ones for the entire sequence step s. Then, by the end of the sequence we use a fitting factor $\mathcal{F}$. L is s's sequence total duration and $\mathcal{F}$ the required length

$$\mathcal{F} = \frac{\mathcal{L}}{L}.$$

If $\mathcal{F} < 1$, we find that the generated sequence duration is higher than the required duration and we multiply all the generated sequence by $\mathcal{F}$. Else if $\mathcal{F} > 1$, it means that the required duration is higher than the sequence matched duration then we need to either repeat step sequence or stretch the durations by multiplying them by a factor $\mathcal{F}$.

We present a recap of the Time Matching stage in Fig. 15.

IV. Predicting Vicinity Dynamics [19]

A lot of studies showed how a node's contact history may be enough to roughly determine future encounters. Beyond this simple knowledge, we observe how convenient it would be to be able to predict pairwise encounters for DTN. This would allow a finer tuning of opportunistic protocols and the possibility to discriminate between different protocol types. We raise the question of the predictability of nodes vicinity behavior. By using the inherent information of the Vicinity Motion model and its transient stochastic knowledge, we expose a heuristic to predict pairwise vicinity distances at future steps. In the DTN research field, knowing which nodes belong to our vicinity and which ones do not is already a helpful point of view. With the κ-vicinity, instead of considering only nodes in contact, we observe neighbors beyond 1-hop. These "close yet not in contact" nodes could be message destinations, or information carriers.

Let us imagine a regular scenario of a daily commuting from home to work. Jean leaves home in the morning takes his favorite commuting mean and heads toward his workplace. At noon, he takes his meal along his coworkers, and then goes back to work. At the end of the day, he leaves his office, returns home and eventually gets groceries on his way back. During his whole daily journey, Jean meets a lot of people, whether he acknowledges them or not. Everywhere Jean moves, people surround him, at home, in the bus, in the streets, at work. Currently, opportunistic networks only gather information about nodes directly around Jean (the 1-hop knowledge). A first way of using Jean's mobility is to observe his current vicinity beyond 1-hop contacts, the more people around him, the more there are potential message carriers. In a previous study, we showed how this simple vicinity observation can help improve performance [7]. During his daily trip, Jean maintains certain regularity. This regularity occurs at various levels. Every morning and every night, he is at home with his family and neighbors. During his commuting, he may travel with the same people whether he realizes it or not, the *familiar stranger* phenomenon [20]. Each workday, Jean interacts with his coworkers. This regularity in meeting patterns can be quite interesting to forward information with smaller costs. Using this potential regularity, we may be able to predict another node's future presence into our κ-vicinity. In our analyses, we forecast pairwise distances between pair of nodes.

A. S3: Vicinity Motion-based Markovian Heuristic

The Markov chain model itself offers a future state prediction model. When we have the average transition probabilities from one state to another of the corresponding AVM, we follow the model evolution to obtain the probability of arriving at any state in the future slotting step. Using transition matrices T and the initial position vector, i.e., at what distance the two nodes are at the beginning, we can infer future steps movement probabilities. Not only can we do it for the future interval/step but also for several steps later.

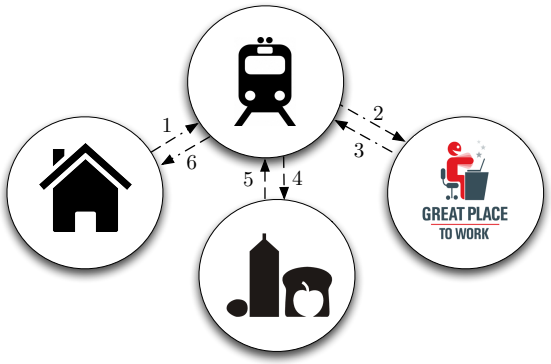


Fig. 16. An example of a workday routine. Jean leaves his house in the morning and takes any public transportation mean (1). He heads to work (2) and stays there with his colleagues all day. At the end of the day, he leaves to get some groceries (3) and (4) and finally heads home for a well deserved rest (6). During his journey, Jean meets a lot of people and visits some key places like the train station, his workplace or his home. In these places, he may meet the same person on and off again. They are part of his vicinity.

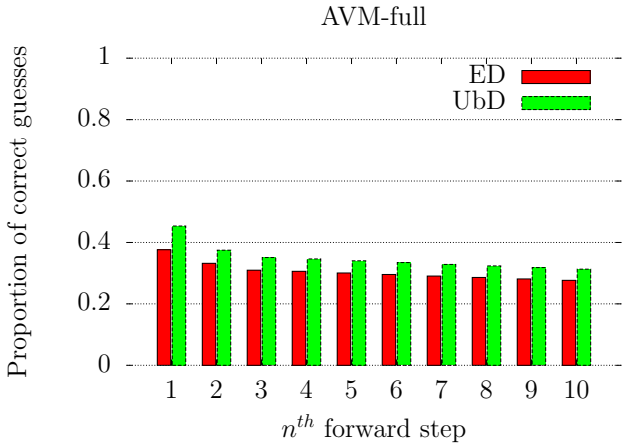


Fig. 17. *Infocom05*'s AVM heuristic performances.

The following heuristic allows us to predict the state i.e., the distance, between two nodes n steps later, based on the current situation. For a given pair of nodes, we apply the position vector to the transition matrix and deduce the probability of being in any state at the future step. This technique provides the probability for the given nodes of being at state $S \in \{\infty, 1, 2, .., \kappa\}$ at the n^{th} future step. The calculus follows:

$$pi_{i,j}(n+m) = pi_{i,j}(m) \times T^n \quad (1)$$

$pi_{i,j}(m)$ is the presence vector indicating the state where two nodes i and j are at step m. For example, given $\kappa = 5$, for the AVM model, the vector [0, 0, 1, 0, 0, 0] indicates that two nodes currently are at a 2-hop distance (state 2). The presence vector [1, 0, 0, 0, 0, 0] shows that the two nodes are in κ-intercontact (state ∞). T^n is the corresponding AVM transition matrix of size $(\kappa+1) \times (\kappa+1)$ to the power $n, n \in \mathbb{N}^*$. $pi_{i,j}(n+m)$ is the probability vector of being at each state $\{\infty, 1, 2, 3, 4, 5\}$ in the following n^{th} step.

Using the resulting $pi_{i,j}(n+m)$ vector, we extract the highest probability state to derive the most plausible state prediction. However, given the nature of opportunistic networking, the connectivity graph is far from fully connected and most of times for the datasets we evaluate, a given pair of nodes is in κ-intercontact (∞). To better detect κ-contact events, we also consider the second highest probability state as a potential prediction. The proposed heuristic outputs two states: the first highest probability S_f and the second one S_s.

The implementation relies on the transitional probabilities provided by the Vicinity Motion module described in S2 and is implemented using Python 2.7 and the NumPy library for matrix calculus.

B. Evaluation

To evaluate the performance of our heuristic, we used the vicinity motion transitional probabilities values and pre-computed the prediction values S_f and S_s for each dataset, any initial position $\leq \kappa$, and n following steps ($n \in \{1, .., 10\}$). This means that for any pairwise initial distance between 1 and κ, our heuristic predicts two potential distance (S_f, S_s) for the n^{th} future step. Then for each dataset and all of their pairwise timelines, we observe the hop sequence. For each hop value, we observe the distance value of the different n^{th} following step and compare them to the corresponding (S_f, S_s) values. In this study, we choose to observe results for the n^{th} future steps with $n \in \{1, .., 10\}$. To evaluate the performance of our heuristic we will use two approaches:

- **Exact distance (`ED`).** If any of the two values (S_f, S_s) match, we consider the prediction to be accurate in an "exact" way. This shows how our heuristic is able to handle an exact distance prediction.
- **Upper bound distance (`UbD`).** If the real hop distance is below or equal to any of the two values (S_f, S_s), we deem the prediction to be accurate in an "upper" bound distance way.

We evaluate the percentage of correct predictions in both `ED` and `UbD` modes.

In Fig. 17, we plot the proportion of accurate heuristic prediction for *Infocom05* which is representative of most datasets except *Rollernet*. On the x-axis, we present the value of the n^{th} step. The y-axis indicates the proportion of accurate guesses our heuristic makes. We test our two evaluation parameters `ED` and `UbD`. For all our datasets, the `ED` metric gives performances between 24% and 42% of correct predictions. Most datasets have their prediction accuracy decrease gradually with higher step values n. But the results decreases of at most 12% between prediction for the next step ($n = 1$) or the 10^{th} next step ($n = 10$). *Rollernet* has a different progression curve with a lower value for the immediate next step ($n = 1$) than for the other steps. However, the sequence of remaining n values have the same evolution as for the other datasets. For `UbD` the prediction accuracy follows the same evolution as `ED`. The only difference being its higher results. On average it is 5% more efficient than `ED` prediction but it is less precise in terms

of distance prediction.

The exact distance prediction is tougher to get right than the upper distance bound. This feels natural as guessing a range is probabilistically easier than guessing an exact value. If we predict a large enough output value for the upper bound distance value, we may encompass the real observed value. A performance evaluation for partial knowledge is available in [4].

V. The Vicinity Package Overview

The Vicinity package contains the required tools to obtain the knowledge to make of most of our the observations detailed in this paper. For further analyses, please refer to [4]. Two of its main parts are: the Asynchronous Vicinity Motion framework who allows vicinity patterns analyses and TiGeR the pairwise vicinity behavior (timeline) generator. These two entities have been designed to function together as the asynchronous vicinity motion framework provides information such as state transitional probabilities and κ-contact durations distributions that are required by the TiGeR module. However, one can also use asynchronous vicinity motion and TiGeR on their own as long as one provides the required inputs. We next provide a few implementation details for both modules.

- **Asynchronous Vicinity Motion (S1).** In this module, we need to recreate the provided network connectivity and compute all shortest distances for all pairs of nodes. To this end, we simulate network connectivity with the Python library NetworkX [21] and make the required arrangements using Python 2.7.
  -Requires: contact trace, number of nodes in the dataset.
  -Provides: vicinity transitional probabilities, interval durations distribution.
- **TiGeR (S2).** The timeline generator module processes extracted dataset characteristics (vicinity transitional probabilities and interval durations distribution) into synthetic vicinity behaviors. We use Python 2.7 for this stage.
  -Requires: transitional probabilities, interval durations distribution, timeline required durations.
  -Provides: synthetic timelines.
- **Vicinity Motion-based Heuristic (S3).** Based on the AVM module, we gather the vicinity transitional probabilities and form the required transition matrix. Then we perform the calculus detailed in Section IV-A using Numpy 1.7.
  -Requires: transitional probabilities matrix, the n^{th} future step value.
  -Provides: S_f, S_s distance prediction values.

The implementation detailed in S1, S2, and S3 is available at the following address: http://vicinity.lip6.fr. The Vicinity module has vicinity motion analyses, the TiGeR generator and an implementation of the Vicinity Motion-based heuristic. It takes as inputs vicinity dynamics in the form of contact traces so it can be applied to any dataset bearing connectivity knowledge.

Acknowledgments

This work has been conducted under the supervision of Marcelo Dias de Amorim from UPMC Sorbonne Universités and in collaboration with Vania Conan from Thales Communications and Security, and Miguel Elias M. Campista from Universidade Federal do Rio de Janeiro.

References

[1] Financial Times by P. Taylor, "Data overload threatens mobile networks," 2012, http://on.ft.com/1aWUNLb.

[2] V. Cerf, "The Internet is for Everyone," RFC 3271 (Informational), Internet Engineering Task Force, April 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3271.txt

[3] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *ACM Special Interest Group on Data Communication Conference (SIGCOMM)*, Karlsruhe, Germany, Aug. 2003.

[4] T. Phe-Neau, "Properties and impact of vicinity in mobile opportunistic networks," Ph.D. dissertation, Université Pierre et Marie Curie, 2014.

[5] T. Phe-Neau, M. Dias de Amorim, and V. Conan, "Vicinity-based DTN Characterization," in *ACM International Workshop on Mobile Opportunistic Networks (MobiOpp)*, Zurich, Switzerland, Mar. 2012.

[6] ——, "Fine-Grained Intercontact Characterization in Disruption-Tolerant Networks," in *IEEE Symposium on Computers and Communication (ISCC)*, Kerkyra, Greece, Jun. 2011.

[7] ——, "The Strength of Vicinity Annexation in Opportunistic Networking," in *IEEE International Workshop on Network Science For Communication Networks (NetSciCom)*, Torino, Italy, Apr. 2013.

[8] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep., 2000.

[9] E. Yoneki and D. Greenfield, "Inferring Significance of Meeting Groups in Human Contact Networks," in *European Conference on Complex Systems*, Lisbon, Portugal, Sep. 2010.

[10] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, New York, NY, USA, Nov. 2008.

[11] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on opportunistic forwarding algorithms," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 606–620, 2007.

[12] A.-K. Pietilänen and C. Diot, "Dissemination in opportunistic social networks: the role of temporal communities," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Hilton Head, South Carolina, USA, 2012, pp. 165–174.

[13] P.-U. Tournoux, J. Leguay, F. Benbadis, J. Whitbeck, V. Conan, and M. D. de Amorim, "Density-aware routing in highly dynamic DTNs: The rollernet case," *IEEE Transactions on Mobile Computing*, vol. 10, pp. 1755–1768, 2011.

[14] A. Galati and C. Greenhalgh, "Human mobility in shopping mall environments," in *ACM International Workshop on Mobile Opportunistic Networks (MobiOpp)*, Pisa, Italy, 2010, pp. 1–7.

[15] S. Gaito, E. Pagani, and G. P. Rossi, "Fine-Grained Tracking of Human Mobility in Dense Scenarios," in *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, Rome, Italy, Jun. 2009.

[16] S. Pal Chaudhuri, J.-Y. Le Boudec, and M. Vojnovic, "Perfect Simulations for Random Trip Mobility Models," in *IEEE International Conference on Computer Communications (INFOCOM)*, Miami, Florida, USA, Aug. 2005.

[17] T. Phe-Neau, M. Dias de Amorim, and V. Conan, "Caractérisation en diptyque de l'intercontact pour les réseaux à connectivité intermittente," in *Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel 2012)*, May 2012.

[18] T. Phe-Neau, M. E. Campista, M. Dias de Amorim, and V. Conan, "Examining Vicinity Dynamics in Opportunistic Networks," in *ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Barcelona, Spain, Nov. 2013.

[19] A. Tatar, T. Phe-Neau, M. Dias de Amorim, V. Conan, and S. Fdida, "Beyond contact predictions in mobile opportunistic networks," in *Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, April 2014.

[20] E. Goodman and E. Paulos, "The Familiar Stranger: Anxiety, Comfort, and Play in Public Places," in *ACM SIGCHI Conference on Human Factors in Computing Systems*, Vienna, Austria, Apr. 2004.

[21] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.