

Traps and Pitfalls of Using Contact Traces in Performance Studies of Opportunistic Networks

Nikodin Ristanovic, George Theodorakopoulos and Jean-Yves Le Boudec
EPFL, School of Computer and Communication Sciences, Switzerland, firstname.lastname@epfl.ch

Abstract—Contact-based simulations are a very popular tool for the analysis of opportunistic networks. They are used for evaluation of networking metrics, for quantifying the effects of infrastructure and for the design of forwarding strategies. However, little evidence exists that the results of such simulations accurately describe the performance of opportunistic networks, as they commonly ignore some important factors (like limited transmission bandwidth) or they rely on assumptions such as infinite user cache sizes.

In order to evaluate this issue, we design a testbed with a real application and real users; we collect application data in addition to the contact traces and compare measured performance to the results of the contact-based simulations. We find that contact-based simulations significantly overestimate delivery ratio, while the captured delay tends to be 2-3 times lower than the experimentally obtained delay. We show that assuming infinite cache sizes leads to misinterpretation of the effects of backbone on an opportunistic network. Finally, we show that contact traces can be used to analytically estimate the delivery ratios and the impact of backbone, through the dependency between a user centrality measure and her delivery ratio.

I. INTRODUCTION

Many performance studies of opportunistic networks are based on contact traces. They are used for estimation of fundamental networking measures, such as delay or delivery ratio [1], [2]. They are also used for the design of caching and forwarding strategies [3], [4] and in the studies of the effects of adding infrastructure to an opportunistic network [5]. Intuitively, contacts between opportunistic users are one of the key factors to take into account when modeling information propagation in an opportunistic network. Nevertheless, contact traces have certain limitations. By default, contact-based studies do not address the limited transmission bandwidth [6], [2]. Traffic generation is artificial or obtained from a distribution [7], [8]. Certain technology limitations, such as the inability of Bluetooth to concurrently discover and send data are ignored. In addition to this, contact-based studies often assume infinite sizes of users' caches and data exchanges without prioritization [1]. This, coupled with the absence of a model for the limited transmission bandwidth, can lead to simulations of unrealistic data exchanges.

In spite of the obvious need to quantify the effects of these approximations, little effort has been invested in justifying the perpetual use of contact traces for the analysis and simulation in the area of opportunistic networks. In other words, little evidence confirms that values obtained from the simulation on contact data sets accurately describe performance of opportunistic applications. In this paper, we perform an initial study

of the often neglected factors and assumptions in the contact-based simulations. Our goal is to find out whether contacts are sufficient to evaluate performance of an opportunistic network, with or without an infrastructural component and if so, how to use them best to achieve this.

The task is far from trivial. The conclusions of contact-based simulations need to be compared against the data coming from the use of real opportunistic applications. As such data is hard to find, we develop an easy to use opportunistic application that can also internetwork with the Internet. We then run an experiment with 50 users, during 2.5 weeks in order to collect contact traces and the application data that can be compared with the results of contact-based simulations.

Instead of inventing a new application, we decided to extend an existing web application - Twitter to the intermittently connected opportunistic space. This significantly simplifies the bootstrapping phase (exploits an already established user base and relationships between users), shortens the learning curve of the experiment participants and allows them to keep their existing habits and use the application in a more natural way.

The choice of Twitter also allows us to cover several realistic use cases. For example, it is quite expensive for roaming users to synchronize their mobile applications with the Internet on foreign networks. However, for a broad set of applications, such as e-mail, Twitter, Facebook and other social-networking apps, the synchronization may not be needed in real time. An opportunistic Twitter application, with occasional access to data sinks that provide Internet connectivity, might deliver tweets with acceptable delays.

Another example, where an opportunistic Twitter application (accompanied with a few points of interconnection with the Internet) can be of great help are deliberate shutdowns of telecommunication networks during protests. Internet blackouts target primarily Twitter and other social networks, with the goal of preventing information propagation and communication among protesters. Solutions, such as voice-to-tweet software provided by Google and Twitter [9] can allow users to tweet using voice. Nevertheless, when the mobile phone service is down, the opportunistic communication, supported by a few satellite Internet connections [10], remains the only option.

We make the following contributions:

- We show that the common practice of ignoring certain factors in the contact-based studies of opportunistic networks significantly affects important performance metrics. Namely, we show that contact-based simulations overestimate delivery ratios up to 30%, while the estimated delays are 2-3 times lower

than the experimental values. We demonstrate how the default assumption in contact-based simulations about the unlimited cache sizes completely alters conclusions about the utility of a backbone in an opportunistic network. We verify the robustness of our findings by rotating the caching strategies.

- We show that weighted contact graph can be a very useful tool for statistical analysis of opportunistic networks. We find a strong dependency between a user centrality measure in this graph and the perceived delivery ratio and we fit a simple curve to this dependency. This allows one to predict users' delivery ratios based on the contact trace. We show that this dependency persists when a backbone is added to the network, which means that it can be used to estimate the effects of adding limited infrastructure to an opportunistic network.

- From the systems aspect, our study gives a comprehensive insight into performance of a small to medium opportunistic network and to a lesser extent into users' reaction to it. It also highlights certain design choices used to extend an existing web application to the world of intermittently connected devices.

The paper is organized as follows. We describe our experiment setup in §II. We introduce notation in §III. We analyze the obtained data sets in §IV. In V we give insights into the experimentally obtained performance and we compare it with the results of the contact-based simulations. We highlight common traps of the simulation based approach. Finally, in §VI we use contact graph for statistical analysis of network performance and prediction of delivery ratios.

II. THE EXPERIMENT SETUP

We design our experiment with two main goals in mind: (i) To collect the application data from which important performance metrics can be extracted and (ii) to collect the contact trace that can be used in discrete event simulations. This will allow us to compare the experiment results with the values obtained from the contact-based simulation.

A. The Experiment Scenario

We use the scenario of roaming users as the running example in our experiment (although the scenario itself is not essential for the results of our study). We assume a mixed population at the university campus site, composed of visitors (Roaming Users) and users in their home networks (Home Users). As policy restrictions prevent Roaming Users (RUs) from connecting to the Internet via the campus WLAN, we assume they prefer using an opportunistic application by paying high roaming fees for data traffic. It is difficult to involve visitors in a rather long experiment. Thus, we chose 50 volunteers to represent the RUs. While fully aware that the mobility of real roaming users can be somewhat different, we find that our experiment participants share certain mobility properties with campus visitors. About half of the participants are master students who followed courses in the classrooms where winter schools are organized (only for visitors). All participants normally have lunch at the same places where visitors are likely to have lunch or coffee.

Home Users (students/faculty) are normally in majority. They have laptops with access to the campus WLAN, and/or

inexpensive data plans with mobile operators. We assume some of them are cooperative and willing to run a piece of software on their devices, helping Roaming Users deliver their tweets to the Internet and receive the tweets of the people they follow. Creating a significant Home User population for the purposes of the experiment (in addition to the Roaming User population we had to recruit) would require substantial financial and human resources. Thus, we resort to an abstraction. We place ten Linux laptops in popular places around the university campus (restaurants, computer rooms, coffee shops, libraries, etc.). We refer to these machines as Home User Equivalents (HUEs). We believe this is a good approximation, as (i) these are the locations where Home Users (with their cell phones and laptops) can be found during the day, (ii) the range of the Bluetooth dongles plugged into HUEs matches the Bluetooth range of cell phones and laptops ($\sim 10m$), and (iii) the set of functions handled by the HUEs is very limited, which means that the code can easily run on any piece of hardware (smartphones, laptops, etc.).

B. System Architecture

Our experimental setup consists of three main parts (Figure 1): (i) Roaming Users (RUs) with the opportunistic Twitter application running on their phones, (ii) Home User Equivalents (HUEs) that serve as interconnection points between the opportunistic space and the Internet, and (iii) our proxy server in charge of communication with the HUEs on the front-end and synchronization with Twitter servers on the back-end.

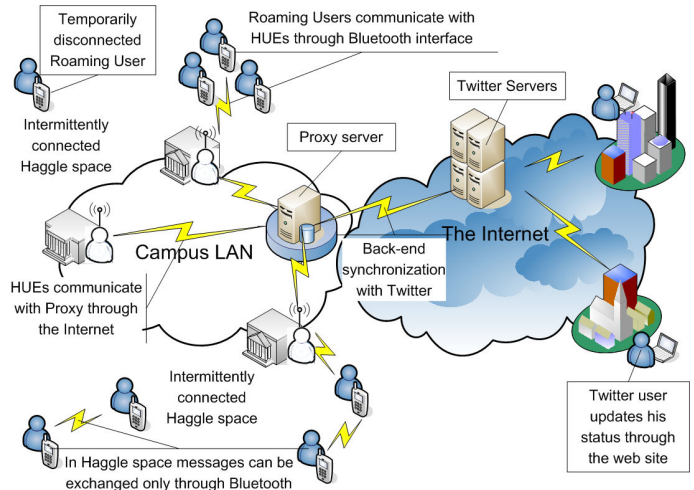


Fig. 1. The system comprises three major components: (i) Proxy server, (ii) Home User Equivalents (HUEs), and (iii) the phones carried by Roaming Users (RUs). Proxy server communicates with Twitter servers at the back-end and with the HUEs at the front-end. HUEs provide Internet connectivity to RUs.

Opportunistic Twitter. Opportunistic Twitter is a mobile Twitter application we developed for the publicly available Huggle publish/subscribe framework [11]. It leverages intermittent Bluetooth connectivity for the exchange of messages with other devices running the Huggle framework. The framework can work on top of several mobile and desktop operating systems (Windows, Android, Linux, Mac OS, etc.). In our

experiment, the framework and the application run on HTC and Samsung phones. Like most Twitter applications, our application allows to a user with Twitter account to follow a set of other users or channels. As a result, messages (“tweets”) created by these channels become visible in the user’s message feed. So, every RU in our experiment has a group of other users/channels that she follows, as well as a group of followers that receive her updates. We refer to these relationships as the “Twitter following relationships”. The tweets created by RUs, as well as the changes in the “Twitter following relationships,” propagate through the network using a form of epidemic, as explained later in this section. Vibration informs users of message reception.

For the discovery of nearby users, all RUs and HUEs use the Bluetooth inquiry mechanism that allows them to find other Bluetooth devices within transmission range. Conducting inquiries consumes power, so one has to be moderate when setting the inquiry interval. Additionally, while inquiring, a device cannot answer other devices’ inquiries, so performing frequent inquiries is not the best solution. On the other hand, choosing a too large interval results in missed discoveries (exchange opportunities). As users can recharge their phones on a daily basis, we choose the inquiry interval of 2 minutes. If a contact that was seen during the previous inquiry disappears during the following inquiry, but reappears again during the subsequent inquiry, we assume that the recorded contact was never broken.

Home User Equivalentents (HUEs). HUEs run a small application on top of Huggle framework and they have Internet connectivity. This allows to the RUs to use them as sinks and have their tweets delivered to the Internet, i.e. to their external followers around the world. HUEs can also fetch content from our proxy server and deliver tweets from the Internet to the RUs inside the campus.

Proxy. Our proxy server is a component of the system that resides between Twitter servers and HUEs. It is a Java Web application running on Apache Tomcat 6 server that uses a MySQL database for storing all the information important for the operation of the system and for the post-experiment data mining process. Given the current restrictions of Twitter API this component is needed to handle user authentication and to serve as a buffer between the Twitter servers and HUEs.

On the back-end, the proxy passes to Twitter servers the tweets that arrive from HUEs. It also fetches from the Internet the tweets of interest to experiment participants, by synchronizing the local copies of their accounts with the accounts on Twitter servers. On the front-end, the proxy processes HTTP requests received from HUEs, it performs the database transactions and it sends back messages that need to be pushed into the opportunistic Huggle space.

C. Caching Strategies

Caching in RUs. Caching strategies (also called replication strategies) determine the channels that a user should store on the device and then forward. Note that *channels*, not packets, are the proper abstraction for Twitter traffic propagation, contrary

to forwarding strategies in opportunistic networks. The reason is that users express their interests by choosing channels to follow. These interests remain relatively stable and they do not change on a packet-by-packet basis.

One can classify caching strategies according to how selfish they are: The more selfish a strategy is, the more preference it gives to channels that the user is interested in. In contrast, the more altruistic a strategy is, the more it prefers channels that are of interest to the rest of the community (network). The choice of strategy can affect network performance metrics.

In our experiment, we want to make sure that our conclusions are robust with respect to the choice of caching strategy, so we use three very different strategies. The first strategy is extremely selfish, storing only channels that the user is subscribed to; the second is extremely altruistic, preferentially storing channels that the user is *not* interested in; the third, which we refer to as proportional strategy (proportional to channel popularity), balances between the two extremes. More specifically, the third strategy [3] always stores the channels that a user is subscribed to and uses the remaining cache space for helping other channels. When two devices meet, each helped channel is a candidate for replacement, and each device performs the following operations: A locally helped channel c is selected uniformly at random among all locally helped channels, and a remote channel c' is selected uniformly at random among those remote channels that are not locally present. Then, the local channel c is dropped and replaced by the remote channel c' with probability $\min\{1, \frac{\beta_{c'}}{\beta_c}\}$, where β_c is the number of users following channel c .

Although considering an altruistic strategy can seem like a strange choice, it is important to understand that the caching strategy can be chosen by someone else, other than the application users. For instance, an application developer can intentionally add a dose of altruism in order to improve the overall performance.

We choose cache sizes for the RUs that we believe are commensurate with the parameters of our experiment, e.g., the number of users (devices), the amount of traffic that they generate, and the device hardware capabilities. Our objective is to examine the effect of a constrained cache size on the performance of the application. If the cache size is large, it will be practically infinite for the purposes of our experiment, so the results would not be representative for a larger network. We present results for cache sizes of 10 and 20 messages. The rationale is to be able to use the obtained results as best-effort indications of the performance in a larger scale deployment. Additionally, cached tweets are aged out after 8 hours, as we assume that older tweets are of no interest to Twitter users.

Caching in HUEs. Home User Equivalentents (HUEs) have Internet connectivity. They can access all tweets available at the proxy in real time. However, keeping all tweets of interest to RUs in HUEs’ caches is unwise. Downloading all these tweets to the local HUEs’ caches, would increase the bandwidth cost for HUEs. Additionally, this approach has a scaling issue with the increase in number of RUs. Thus, we make the content

available at HUEs adapted to the context, i.e. to the interests of RUs in the vicinity of HUEs and other RUs that can be reached in the near future. HUEs have caches of 40 messages and they are refreshed upon reception of messages from RUs and messages pushed by the proxy.

D. Putting it All Together

Message Flow. The users create tweets that are forwarded among them in the following way: Upon a meeting between two users, messages in their caches are exchanged over Bluetooth. The Huggle pub/sub framework prioritizes message exchanges according to user interests: Messages of higher interest will be exchanged first, followed by the remaining messages in the cache of the other user’s device. This prioritization is crucial when contacts are too short to exchange all messages of both caches. After the exchanges are over, the local caching strategy decides which messages, if any, should be dropped. To avoid transmitting messages that are then dropped, we align the Huggle prioritization with the caching strategy used at the time.

The HUEs are interconnection points between the Internet and the disconnected Huggle space. The reception of a message from a RU triggers creation of an HTTP request by the HUE that is sent to the proxy through the Internet. The proxy processes the request, performs necessary transactions with the database and returns a set of messages (“tweets”) as response. The HUE adds these messages to its local cache and makes them available to Huggle devices in its vicinity.

Experiment Population. Our RUs’ population counted 50 people. Most of them received phones with the opportunistic Twitter application; some of them used their own phones. For the rest of the paper, we will be referring to our population of Roaming Users (RUs) also as *internal users*. Many participants continued using their existing Twitter accounts. The others were free to choose the channels to follow. A followed channel can be either internal (content created by an internal user) or external (content created by an arbitrary Twitter user on the Internet, henceforth collectively called *external users (or channels)*). The social graph obtained from the “Twitter following relationships” shows that almost all internal users follow some internal and external channels. As the content created by external users is also propagated in our system we can, in a way, consider the external users as a part of the experiment.

III. NOTATION AND METRICS

Let $\mathcal{N} = \{1, \dots, N\}$ be the set of internal users, let $\mathcal{X} = \{1, \dots, X\}$ be the set of external users, and let $\mathcal{F}_j \subseteq \mathcal{N} \cup \mathcal{X}$ be the set of users that user $j \in \mathcal{N}$ follows.

Let $\mathcal{A}, \mathcal{B} \subseteq \mathcal{N} \cup \mathcal{X}$ be arbitrary subsets of users. We use $\mathcal{M}_{\mathcal{A} \rightarrow}, M_{\mathcal{A} \rightarrow}$ for the set and number of messages generated by any user $i \in \mathcal{A}$; $\mathcal{M}_{\rightarrow \mathcal{B}}, M_{\rightarrow \mathcal{B}}$ for the set and number of messages delivered to any user $j \in \mathcal{B}$, and $\mathcal{M}_{\mathcal{A} \rightarrow \mathcal{B}} = \mathcal{M}_{\mathcal{A} \rightarrow} \cap \mathcal{M}_{\rightarrow \mathcal{B}}$. Only the messages generated by users that j follows can ever be considered to be “delivered” to j , but not the messages that j receives just to forward on behalf of others.

For an internal user $j \in \mathcal{N}$ and a message $m \in \mathcal{M}_{\rightarrow j}$, let D_j^m be the delivery delay of message m to user j . That is, D_j^m

is the time elapsed between the generation of m at some user $i \in \mathcal{F}_j$ and the delivery of m to j .

For internal users $j \in \mathcal{N}$ we define the following metrics: The *delivery ratio* $R_j^{\mathcal{A}}$ from \mathcal{A} to j is the fraction of messages generated by users in \mathcal{A} and delivered to user j over the total number of messages generated by users in \mathcal{A} and destined for user j .

$$R_j^{\mathcal{A}} = \frac{M_{\mathcal{A} \rightarrow j}}{M_{\mathcal{A} \cap \mathcal{F}_j \rightarrow}}. \quad (1)$$

When $\mathcal{A} = \mathcal{F}_j$ we drop \mathcal{A} from $R_j^{\mathcal{A}}$, and we simply call R_j the *delivery ratio*; $R_j^{\mathcal{N}}$ is the *internal delivery ratio*, and $R_j^{\mathcal{X}}$ is the *external delivery ratio*.

We define the *message delay* $D_j^{\mathcal{A}}$ from \mathcal{A} to j as the average delay over all messages generated by users in \mathcal{A} and delivered to user j .

$$D_j^{\mathcal{A}} = \frac{\sum_{m \in \mathcal{M}_{\mathcal{A} \rightarrow j}} D_j^m}{M_{\mathcal{A} \rightarrow j}} \quad (2)$$

As with the delivery ratio, we call D_j the *message delay*; $D_j^{\mathcal{N}}$ is the *internal message delay*, and $D_j^{\mathcal{X}}$ is the *external message delay*.

We are also interested in evaluating the quality of the synchronization between the opportunistic part and the Internet part of the application. For this purpose, we treat the Proxy server as another user and measure its delivery ratio and message delay. We use the same two definitions as for mobile users, but we assume that the Proxy follows all internal users.

IV. OBTAINED DATA SETS

As a result of the experiment we get two data sets: (i) the application metadata that we use to extract the fundamental performance metrics, such as delay and delivery ratio and (ii) the contact trace, which we use in trace driven simulations to obtain the same metrics from contacts. Each of the three caching strategies applied at RUs is evaluated for two different cache sizes: 10 and 20 messages. This gives a total of six combinations, each of which is tested during two working days.

In the trace driven simulations that we perform after the experiment we implement the same combinations of caching strategies and cache sizes. Each combination is simulated using the corresponding 2-day contact trace.

In our experiment, an average internal user (RU) follows 9 internal and 14 external channels (> 600 external channels in total). The maximum number of internal and external channels followed by an internal user are 17 and 98, respectively. The most popular internal channel is followed by 18 internal users, while the most popular external channel has 8 internal followers. The internal users alone create 3010 tweets during the experiment. “Twitter following relationships” between internal users are shown in Figure 2.

Figure 3 shows the total number of contacts each of the 50 internal users and 10 HUEs have with other internal users and HUEs, during the six observed 2-day periods. We distinguish between contacts with followed internal users, with other internal users and with HUEs. The total number of contacts varies depending on user ID and the day of the week. For example, the

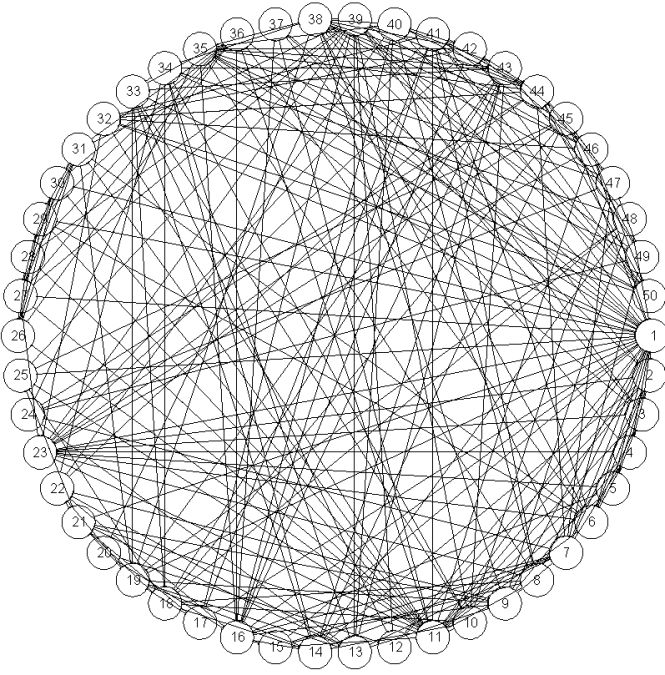


Fig. 2. “Twitter following relationships” between internal users. An edge between two users means that one of the users follows the other one.

students of the Master’s program have lectures and labs together on Thursdays and Fridays. Subfigures 1, 2, 4 and 6 (from top left to bottom right), which correspond to 2-day periods that contain either Thursday, Friday or both, clearly show more contacts (116, 98, 123 and 116 contacts per user per day, respectively) than subfigures 3 and 5 (56 and 59 contacts per user per day), which correspond to combinations of Mondays, Tuesdays and Wednesdays. The contact durations follow a similar pattern. For this reason, the comparison between the caching strategies is not perfect, but a more comprehensive study on this topic is out of the scope of this paper.

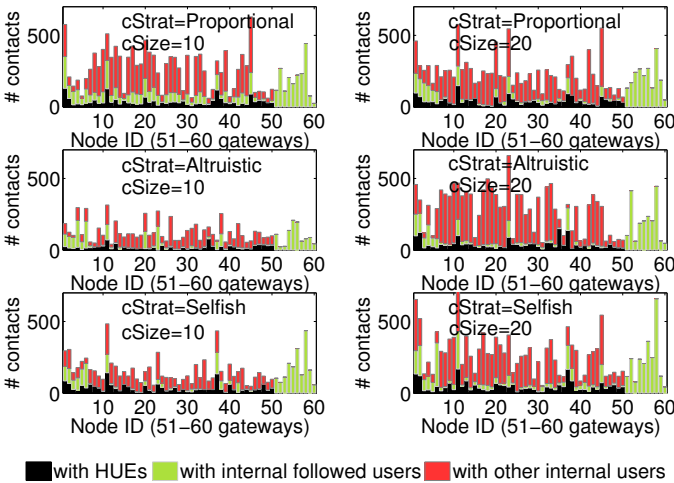


Fig. 3. Total number of contacts experienced by internal users ($j = 1, \dots, 50$) and HUEs ($j = 51, \dots, 60$) during 2-day evaluations of the 6 combinations of caching strategy $cStrat$ and cache size $cSize$.

In Figure 4 we plot the CCDF of inter-contact times between the internal users and HUEs. It shows how often RUs visit the popular locations within the campus where HUs can be found. We view all HUEs as parts of the same backbone (as they all have access to the Internet) and calculate inter-contact times with it for all internal users. The CCDF in Figure 4 is flat between 3 and 20 hours, which implies that it is more probable for a user to meet a HUE soon after the previous meeting. We also see that 80% of inter-contact times with HUEs is shorter than 50 min and only 3% is longer than 24h. We observe two drops, at 20-25 hours and at 3 days, corresponding to meetings that happen once a day around the same time, and Friday meetings that happen again on Mondays.

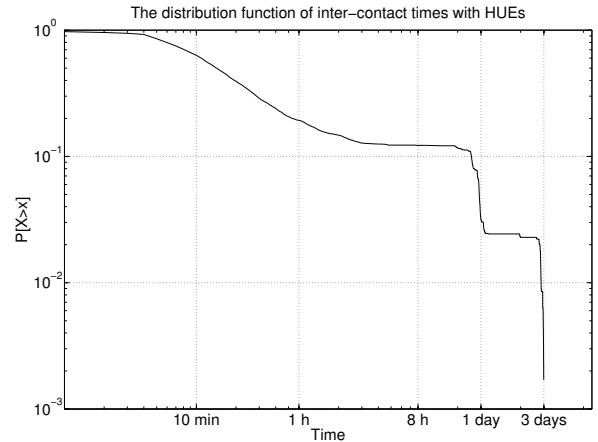


Fig. 4. The distribution function of inter-contact times with HUEs obtained for the whole duration of the experiment.

V. PERFORMANCE METRICS: EXPERIMENT RESULTS VS. CONTACT-BASED SIMULATION

The availability of the application metadata and contacts for the same experiment allows us to test the accuracy of commonly used contact-based simulations. The rotation of caching strategies permits us to verify the robustness of our conclusions. We focus on two fundamental networking measures, namely, delay and delivery ratio. In the case of both metrics we first analyze the values obtained from the experiment. We then compare these values with the corresponding values obtained from the contact-based simulations. Finally, we study the effects of adding a backbone to an opportunistic network, showing that as a rule, contact-based studies underestimate the impact of backbone, due to hidden assumptions.

A. Delivery Ratio: Experiment vs. Contact Simulation

Experimentally obtained delivery ratios. Figure 5 shows the internal and external delivery ratios, R_j^N and R_j^X , seen by internal users ($j = 1, \dots, 50$) and by the proxy ($j = 51$) during the observed evaluation periods. Each period of two working days corresponds to a combination of a caching strategy and a cache size. We see that proportional strategy performs on average 10-20% better for both evaluated cache sizes. We observe

higher delivery ratios when the cache size is 20, regardless of the caching strategy. Finally, through the performance of user 51 (proxy), we see that almost all messages, created by internal users are delivered to Twitter web site.

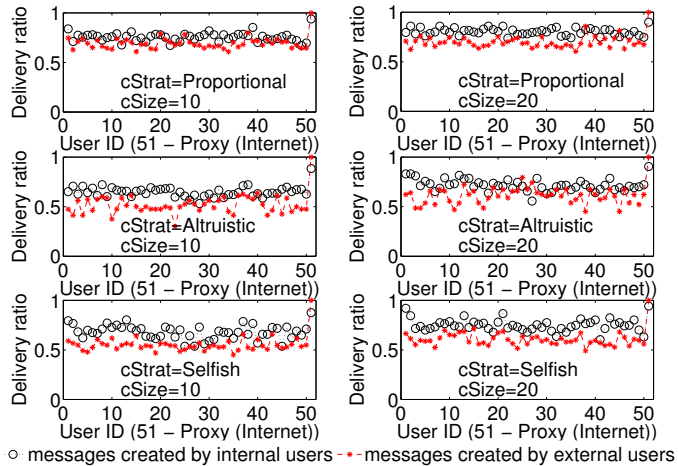


Fig. 5. Internal and external delivery ratios, R_j^N and R_j^X , seen by the internal users ($j = 1, \dots, 50$) and by the proxy ($j = 51$). Every combination of caching strategy ($cStrat$) and cache size ($cSize$) was evaluated during 2 days.

Figure 5 also shows that the external delivery ratio is lower than the internal. The reason is that the number of external channels is large (> 600) and there is only a limited overlap between channels followed by internal users. So, each cached external channel is useful to few internal users. Even if caches were full of external channels, there would still be channels that are not cached anywhere, thus making it difficult for the followers of these channels to receive them.

Contact simulations overestimate delivery ratios. We now compare the experimental results with the values of delivery ratio obtained from the contact-based simulations. Each combination of cache size and caching strategy is simulated using contacts collected during the period when the combination was used. In Figure 6, the full lines on the right subfigure represent delivery ratios perceived by experiment participants, for messages created by internal users and for cache size of 20 messages. The same unsorted values are shown on Figure 5 (three subfigures on the right). The left subfigure on Figure 6 contains the corresponding delivery ratios, obtained from the contact-based simulations of the same caching strategies for the cache size of 20 messages. It also contains delivery ratios obtained from the simulation with unlimited cache sizes, where users cache all received messages (top full line). This is the most represented case in the existing literature [2], [1].

The two subfigures allow us to draw the first two conclusions about the deficiencies of contact-based simulations. First, contact-based simulations overestimate the delivery ratios. This is due to the fact that they fail to model the limited contact durations and transfer bandwidth, as well as the limitations of the used wireless technology. In other words, some recorded contacts do not result in transfers and some of them allow transfers of only a part of available data. This is further

confirmed in Section V-B, where we analyze delays. Second, assuming unlimited cache sizes always increases delivery ratios. For example, we can see that this assumption increases delivery ratios for up to 30%, compared to the case with altruistic caching strategy and the cache size of 20 messages.

Misinterpreting the importance of backbone. The collected data set enables us to study the improvement that a backbone brings to opportunistic communication. The application metadata allows us to differentiate between copies of a message that traversed the backbone (HUEs, proxy) in the process of forwarding and those that reached their destinations using pure ad hoc forwarding. By considering the former as lost, we calculate delivery ratios and delay in a hypothetical system without backbone connectivity. As external messages cannot enter the system without the backbone, the metrics in the hypothetical system are about internal messages only. Similarly to the definition of R_j in Section III, we define R'_j as the fraction of messages delivered to user j over the total number of messages destined for user j , in a system without a backbone.

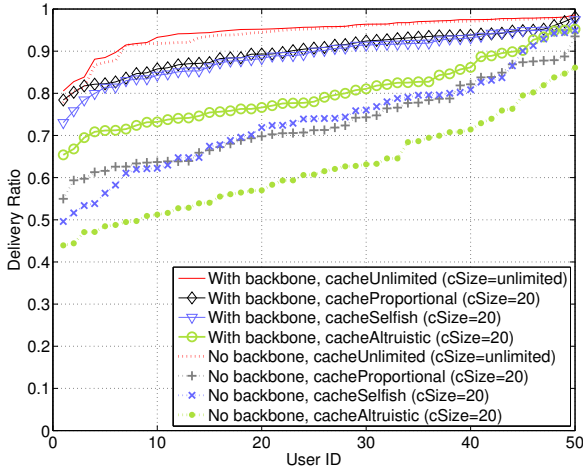
The dotted lines in Figure 6 represent delivery ratios in the system without backbone (HUEs, proxy), for different caching strategies and the cache size of 20 messages. Again, the contact-based simulation significantly overestimates delivery ratios (about 30% in the case of proportional caching strategy).

Figure 6 allows us to observe another trap of contact-based simulations. We see that in the case of limited cache sizes backbone brings significant improvement to delivery ratios. However, in the comprehensive simulation study in [1] the authors conclude that backbone brings only marginal improvement to delivery ratios. This conclusion is the result of an often hidden assumption in contact based studies that cache sizes are infinite. Indeed, as we see in Figure 6, in the simulated case with unlimited cache sizes, backbone brings almost negligible improvement. This is due to the fact that a user with unlimited cache can store much more information, so during a contact, she can provide almost as much data as a backbone.

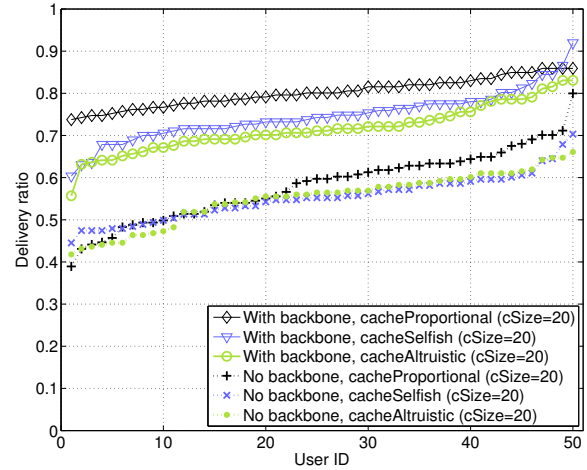
B. Delay: Experiment vs. Contact Simulation

Delay obtained from the experiment. Figure 7 shows the internal and external message delays, D_j^N and D_j^X , observed by the internal users ($j = 1, \dots, 50$) and by the proxy ($j = 51$). The average internal delay typically ranges from 100 to 140 minutes. The average external delay is higher. Intuitively, one would expect the external messages to reach their destinations faster, due to their availability at all HUEs soon after creation. Messages created by internal users, in contrast, experience a non-negligible delay before becoming available at HUEs, as we can see from the delay observed by the proxy ($j = 51$ in Figure 7). However, as we observe in our message log, some of the external messages created in different time zones are created during the night. This introduces delay, as there are very few or no internal users on the campus in the nighttime.

Contact-based simulation underestimates delay. In Figure 8 we plot delays obtained from the experiment and contact-based simulations, for the cases with and without backbone. We see that simulations give delays that are 2-3 times lower



(a) Delivery ratios - simulation



(b) Delivery ratios - experiment

Fig. 6. Delivery ratios obtained from the simulations and from the experiment for different caching strategies. The full lines correspond to the system with the backbone (HUES, proxy), while the dotted lines describe the system without the backbone.

than the experimentally obtained delays. We inspect the contact trace and the application data and we observe that recorded contacts do not always result in message transfers. This means that limited transmission bandwidth, short contact durations and inability of Bluetooth to concurrently scan and send data prevented users from leveraging all transfer opportunities. As most of these limitations are not inherent only to Bluetooth, we conclude that delays obtained from contact simulations should be taken with a grain of salt, as they tend to be too optimistic.

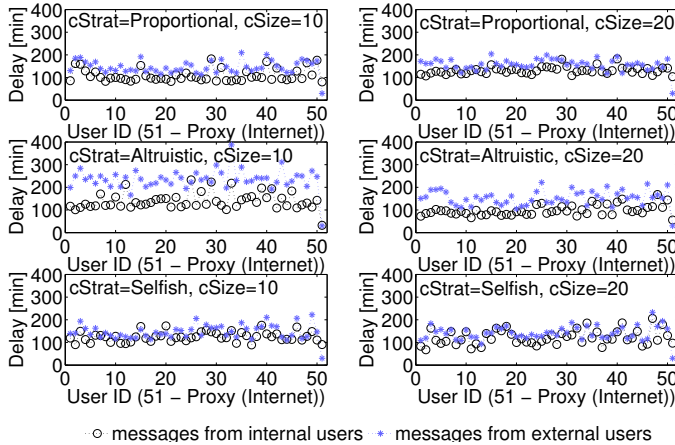


Fig. 7. The average age of received message observed by internal users. Every combination of caching strategy ($cStrat$) and cache size ($cSize$) was evaluated during 2 working days.

Delay from Users' Viewpoint. The recorded delays give us some insights into performance from the networking perspective. However, we would like to know more about users' perception of this performance and their reaction to it. Twitter option called “@replies” allows us to find out more about this.

When a Twitter user receives a tweet she wants to respond to,

she can create an @reply message, by putting @ + the name of the creator of the original tweet in his reply. This helps us easily identify pairs containing original tweets and @replies to these tweets. We then record delays for the tweets whose reception led to the creation of @replies by the recipients and we plot the corresponding CCDF (Figure 9). We see from the figure that 60% of the tweets that receive an @reply are received with a delay inferior to 2h. However, 40% of the tweets that instigated the creation of an @reply message are received with a delay between 2 and 3h, which means that the recipients still find this non-negligible delay acceptable. In addition to this, we find that many of the @replies are threaded and parts of longer conversations (we also verify this by checking the message content), which means that the observed delays allow users to maintain longer message exchanges.

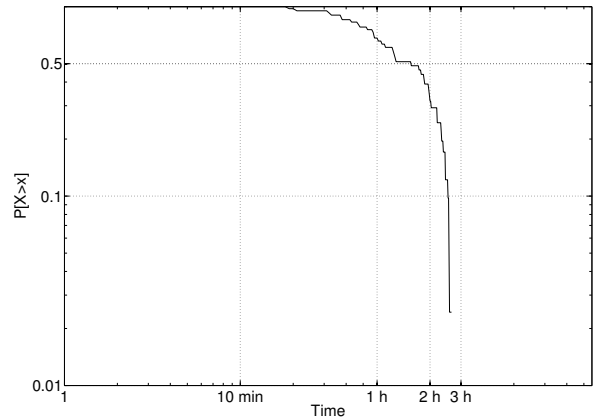
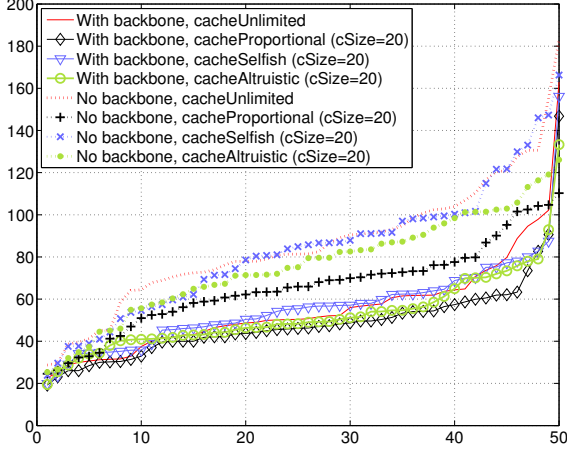
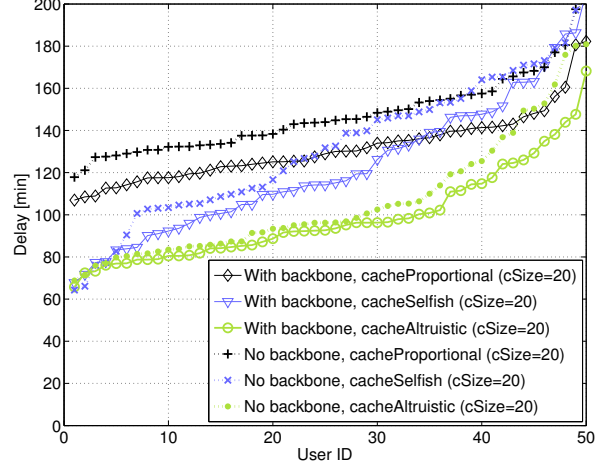


Fig. 9. The distribution function of observed delays for the tweets whose reception led to the creation of @replies by the recipients.



(a) Delays - simulation



(b) Delays - experiment

Fig. 8. Delays obtained from the simulations and from the experiment for different caching strategies. The full lines correspond to the system with the backbone, while the dotted lines describe the system without the backbone. The case with unlimited caches is also simulated.

VI. CLOSENESS CENTRALITY PREDICTS DELIVERY RATIO

In Section V we show that simulations on contact traces suffer from multiple drawbacks. A contact trace can also be analyzed using its statistical properties. The goal is the same, estimating the performance of a network/application. We apply the following approach: to represent the contacts among users, we define the *contact graph* as an undirected weighted complete graph $G_{con} = (\mathcal{N} \cup \{I\}, E_{con})$. The vertex set comprises the internal users and the vertex I representing the infrastructure. As the graph is complete, the edge set E_{con} comprises all unordered pairs of vertices. The weight of the edge $ij \in E_{con}$ is equal to $w_{ij} = \frac{1}{c_{ij}^\lambda}$, where c_{ij} is the number of contacts between users i and j , and λ is a real number constant.

In the graph G_{con} , we denote by $d_{ij}(\lambda)$ the shortest path distance between i and j . The *average shortest distance* $d_i(\lambda)$ of a node i (other than I) to all other nodes in the graph is

$$d_i(\lambda) = \frac{\sum_{j \in \mathcal{N} \setminus \{i\} \cup \{I\}} d_{ij}(\lambda)}{N}, \quad (3)$$

also called *closeness centrality* in the social network literature [12]. The lower this quantity is, the more connected a node is. We find a noticeable dependency between the delivery ratio R_i of a node i and the node's closeness centrality d_i . In particular, the following curve fits the data well:

$$R_i = \frac{1}{1 + kd_i(\lambda)}, \quad \lambda = 0.95, \quad (4)$$

where k is a constant that depends on the caching strategy (discussed later). Other centrality measures that we tested, namely degree centrality, eigenvector centrality and betweenness centrality, result in weaker dependency. By applying a similar approach to the social graph shown in Figure 2, we find no dependency between delivery ratio (or delay) and centrality measures in this graph.

The weight exponent λ can change the relative importance of small and large edge weights. The weight of a path p is the sum of the weights of its edges e_1, e_2, \dots, e_l :

$$w(p) = \frac{1}{c_{e_1}^\lambda} + \frac{1}{c_{e_2}^\lambda} + \dots + \frac{1}{c_{e_l}^\lambda}. \quad (5)$$

With a large positive value of λ , the edges with a small number of contacts dominate, whereas with a large negative value of λ , the edges with a large number of contacts dominate.

We choose $\lambda = 0.95$ because this value maximizes the mutual information between $d(\lambda)$ and R , viewed as discrete random variables. Intuitively, the mutual information of $d(\lambda)$ and R is high when the knowledge of one reduces our uncertainty about the other, which is desirable as we want to use d to predict R . The advantage of using mutual information as opposed to, for instance, correlation, is that mutual information is not biased by the relative values of the quantities involved. So, we see that, to maximize the predictive power of d for R , all edge weights should be treated with approximately equal importance. After choosing $\lambda = 0.95$, we do curve fitting to find the value of k that minimizes the sum of vertical distances. The values of k are always in the interval $[2.7, 3.5]$.

Knowing the dependency and using the curve helps one estimate a typical node's expected delivery ratio if one can estimate or guess a typical node's closeness centrality. Furthermore, one can form an expectation about the effect of connecting to the backbone (thus changing nodes' closeness centralities) on the delivery ratio that network users will experience.

For $k = 3.1$, we plot the data and the curve in Figure 10. In every subfigure each user's R_i and d_i are plotted for the cases with and without backbone. We see that the $R - d$ dependency holds, not only across users within the same network topology, but persists even across qualitative changes in the topology.

Moreover, we see that the dependency exists, and the curve

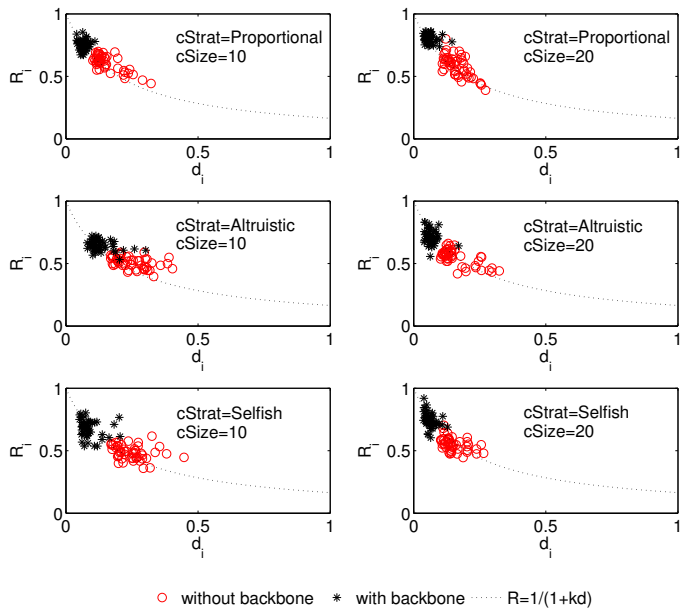


Fig. 10. Dependency between delivery ratio R_i and closeness centrality d_i .

fits, *regardless* of the caching strategy used. This last point is important, because caching strategies affect delivery ratios. But the effect of the caching strategy can be included in the constant k , which is limited to a small range of values. We conclude that a node's distance is a reliable indicator of its delivery ratio.

VII. RELATED WORK

The validation of simulation results with measurements has been used to evaluate the accuracy and determine the level of fidelity of simulation models [13], [14]. To the best of our knowledge, this is the first paper that studies limitations of the contact-based simulations in opportunistic networks. This is somewhat surprising, given the variety of topics and proposals validated using contact-based simulation. A possible explanation can be sought in the cost, scale and complexity of the experimental evaluation, needed for such a study. Although the first of its kind, this paper is closely related to a large body of work that covers various aspects of opportunistic communication through contacts. It concerns contact-based evaluations of caching and replication schemes [3], validations of forwarding protocols [4] and studies of content dissemination in urban environment [2], [7].

Finally, this paper is closely related to the studies of the effect of backbone on opportunistic communication. Initially, they relied exclusively on contact traces [5], [1]. In [1] the authors perform extensive simulations on Bluetooth contacts in order to quantify the effect of the opportunistic and backbone components in a DTN. They conclude that backbone brings only marginal improvement to opportunistic communication. The UMass DieselNet testbed addressed a similar topic, but the Wi-Fi equipped buses exchanged traffic (obtained from the Poisson distribution). The authors observe higher utility of the backbone component [8]. Our study permits to reveal that

much of this discrepancy, in the observed backbone-induced improvement, comes from a common assumption in contact-based simulations about the infinite cache sizes.

VIII. CONCLUSIONS

Several important conclusions can be drawn from this study of the drawbacks of contact-based simulations in opportunistic networks. First, our experimental results show that the commonly ignored factors in simulation studies, such as technology limitations and transmission bandwidth, lead to significant discrepancies between experimental and simulation values. All caching strategies and cache sizes, tested by 50 users during the 2.5 week experiment, unanimously confirm that contact-based simulations tend to overestimate network performance (especially delay). This means that an effort should be made to include these factors in the future trace driven simulations.

Second, we find that some commonly hidden assumptions, like the assumption about infinite cache sizes, result in the overly pessimistic conclusions about the utility of a backbone in an opportunistic network. This is an interesting finding that could direct more attention towards hybrid networks, that include both, the opportunistic and the infrastructural component.

Finally, we show that a statistical treatment of the contact trace, offers a good prediction of certain performance aspects, namely delivery ratio. We show how the existence of a backbone increases the message delivery ratio by reducing user distances on the contact graph. The strong statistical dependency that we find (between node's centrality and delivery ratio) can help predict not only delivery ratios, but also the effect of adding a backbone to an opportunistic network.

REFERENCES

- [1] P. Hui and A. Lindgren, "Phase transitions of opportunistic communication," in *CHANTS '08*.
- [2] J. Leguay, A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft, "Opportunistic content distribution in an urban setting," in *CHANTS '06*.
- [3] L. Hu, J. Le Boudec, and M. Vojnovic, "Optimal Channel Choice for Collaborative Ad-Hoc Dissemination," in *Proc. INFOCOM*, 2010.
- [4] L. Song and D. F. Kotz, "Evaluating opportunistic routing protocols with large realistic contact traces," in *CHANTS '07*.
- [5] A. Lindgren, C. Diot, and J. Scott, "Impact of communication infrastructure on forwarding in pocket switched networks," in *CHANTS '06*.
- [6] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on the design of opportunistic forwarding algorithms," in *Proc. IEEE Infocom*, 2006.
- [7] A. Chaintreau, J.-Y. Le Boudec, and N. Ristanovic, "The age of gossip: spatial mean field regime," in *Proc. SIGMETRICS '09*.
- [8] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Enhancing interactive web applications in hybrid networks," in *Proc. MobiCom '08*.
- [9] Guardian, "Voice-to-tweet," <http://www.guardian.co.uk/technology/2011/feb/01/google-twitter-egypt>.
- [10] N. Y. Times, "Shadow Internet," http://www.nytimes.com/2011/06/12/world/12internet.html?_r=3&hp.
- [11] E. Nordström, P. Gunningberg, and C. Rohner, "A search-based network architecture for mobile devices," tR 2009-003, Uppsala University.
- [12] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1978-1979.
- [13] M. Weigle, "Improving confidence in network simulations," in *Simulation Conference, 2006. WSC 06. Proceedings of the Winter*, 2006.
- [14] G. Lazarou, V. Frost, J. Evans, and D. Niehaus, "Using measurements to validate simulation models of tcp/ip over high speed atm wide area networks," in *Communications, IEEE*, 1996.