Twitter in Disaster Mode:

Opportunistic Communication and Distribution of Sensor Data in Emergencies

Theus Hossmann, Franck Legendre, Paolo Carta
Communication Systems Group
ETH Zurich, Switzerland
lastname@tik.ee.ethz.ch

Per Gunningberg, Christian Rohner
Communication Research Group
Uppsala University, Sweden
firstname.lastname@it.uu.se

ABSTRACT

Recent events (earthquakes, floods, etc.) have shown that users heavily rely on online social networks (OSN) to communicate and organize during disasters and in their aftermath. In this paper, we discuss what features could be added to OSN apps for smart phones – for the example of Twitter – to make them even more useful for disaster situations. In particular, we consider cases where the fixed communication infrastructure is partially or totally wiped out and propose to equip regular Twitter apps with a *disaster mode*. The disaster mode relies on opportunistic communication and epidemic spreading of Tweets from phone to phone. Such "disaster-ready" applications would allow to resume (although limited) communication instantaneously and help distressed people to self-organize until regular communication networks are functioning again, or, temporary emergency communication infrastructure is installed.

We argue why we believe that Twitter with its simplicity and versatile features (e.g., retweet and hashtag) is a good platform to support a variety of different situations and present *Twimight*, our disaster ready Twitter application. In addition, we propose Twimight as a platform for disseminating sensor data providing information such as locations of drinkable water sources. Eventually, we propose to rely on interest matching to scale Twitter hashtag-based searches in an opportunistic environment. The combination of these features make our opportunistic Twitter the ideal emergency kit in situations of disasters. We discuss and define the main implementation and research challenges (both technical and non-technical).

1. INTRODUCTION

Due to their wide distribution, online social networks (OSN) like Facebook and Twitter have recently been adopted by their users as communication tools in times of disasters, both natural and man-made. Not only were they used for sharing information and providing instant updates to the rest of the world, but they were also found effective in connecting and organizing the people directly affected by the disaster [1]. However, today's most wide spread OSNs were not explicitly designed to support disaster communications and lack some features which could improve their value in such situations.

In particular, we here consider situations where communication network infrastructure is partially or totally destroyed – for example by natural forces like earthquakes or floods. Such breakdowns of cellular and fixed data services render OSNs, which require connectivity to the servers, completely useless. Temporary deployments of wireless mesh networks [2] have been proposed for disaster relief. However, the nodes for building a wireless mesh networks have to be distributed first and then deployed. This requires access of an emergency response team to the disaster area. We can imagine that such an operation, even if the emergency response forces are well equipped and trained, takes at least a few hours (in case of difficult terrain maybe even days) to execute. Also, it is difficult to get a complete coverage without careful planning. Hence, during these first hours after a disaster – the most critical time in which many lives can be saved – people are left on their own, without any means of communication to organize themselves. Using satellite communications, suffers from the same setup time since satellite phones are not usually readily available to regular people in disasters.

Other researchers have proposed the use of delay tolerant, ad hoc networks [3, 4] as a means of communication in disaster response. Building on these ideas, we present in this paper a very practical approach for augmenting the users' smart phones to emergency kits: Extending the mobile online social networking apps with a special *disaster mode*. Since most smart phone users have already installed such applications, this extension would make a simple yet effective communication solution which could be used immediately after the breakdown of network infrastructure. The disaster mode is simply an additional feature which can be enabled by the user to start *opportunistic communication* [5, 6]. Instead of relying on existing opportunistic protocol stacks (DTN2 [7], Haggle [5], (PodNet [6]), we decided to rethink the design from scratch from an application-oriented point of view in a specific context, i.e. Twitter in disaster situations. This gives us more flexibility but, by no means, means that we will not borrow principles or features from these stacks.

Twitter, in the recent past, has proven to be a highly useful communication platform in emergency situations [1]. We believe that there are three main reasons why Twitter is particularly well suited for such situations:

1) While Facebook (and other OSNs) focus on connecting users to their friends in relatively narrow circles of users, Twitter is by default open and public: Tweets can be accessed by anyone, not only friends, via the Twitter search feature or by searching or following a *hashtag* (a keyword marked with a preceding #). This fosters communication among users interested in a certain topic (e.g., a specific natural disaster) even *across* circles of users. 2) Twitter's

ExtremeCom2011, September 26-30, 2011, Manaus, Brazil.

simplicity (for example messages are restricted to 140 characters at a time) make it a versatile platform for many different situations. Twitter's evolution has shown that users are creative in adapting the service for different needs e.g., the invention of hashtags, retweets, URL shortening services, etc. However, it is difficult to foresee the exact users' needs for a certain usage, hence, a flexible service is beneficial. 3) The service is widely used already today, and user numbers are increasing fast.

We have extended and implemented a disaster mode for an open-source Twitter client for smart phones.

Summarizing, our contributions are the following:

- We discuss in detail a potential disaster scenario (Section 2) and derive from it the most important user needs in situation of distress.
- We describe the design and prototype of *Twimight*, our disaster Twitter client. It behaves like a normal Twitter client but with a "disaster mode", enabling opportunistic communication where tweets spread epidemically (Section 3).
- In addition to the Twitter disaster mode, we propose to use Twitter as a vector for disseminating sensor data about the user and the environment (Section 4).
- An interest matching service to enable more efficient spreading of tweets based on hashtags is described (Section 4).

Further, we discuss technical as well as non-technical challenges that still need to be addressed in light with state-of-the-art networking research (Section 5).

2. SCENARIO AND USER NEEDS

In the following we set the scene of an imaginary disaster at large, and then specifically from the point of view of a victim. We try to foresee how opportunistic networking could support communication and sensor data dissemination in such a situation. From the scenario, we derive the basic user needs that must be fulfilled by our Twitter client.

Assume a disaster scenario, as it happened in the Fukushima region in spring 2011, where a huge earthquake was followed by a tsunami. Such a disaster leaves thousands of deaths and ten times more homeless. Family members are separated and isolated from each other. Some of them escaped to safe places while others may be stuck in demolished buildings.

The fixed telecommunication infrastructure is fully or partially wiped out, including the cellular systems and 802.11 access points. The power infrastructure may as well be completely down. Roads and rails are closed down or only usable for emergency vehicles. The surviving infrastructure quickly becomes congested by the information storm that follows on a disaster and may run into a halt.

Distressed people try to comprehend the disaster and to find their family and friends. Relatives outside worry about the wellbeing of their family members. Emergency organizations try to organize first aid. Media want to report from inside the disaster area. In short, there is a need for an enormous acute information exchange.

A survivor of a disaster will find himself, in the best case, with a mobile phone but without any cellular coverage. The social networks Facebook and Twitter, used for keeping in contact with families and friends require infrastructure communication networks and do not work. Yet, there is battery energy in the phone that can be used for communication with other phones over WiFi and Bluetooth in the local area. Also, the GPS positioning is working and people can therefore obtain their exact position.

In our scenario, Mr Survive, activates the disaster mode on his mobile smart phone. The phone detects that there is no infrastructure and therefore switches on opportunistic networking via WiFi, Bluetooth and cellular. Social media now run in distributed mode without server support. Mr Survive first wants to find his family members and therefore sends tweets asking about the location of his relatives, geo-tagged with his current position. Such messages are opportunistically exchanged with other people he meets in the local area over Bluetooth or WiFi. If a recipient of a tweet has relevant information (e.g., received tweets from Mr Survive's family members) she replies to the initial tweet to help reunite them. Mr Survive will do the same for others. In our scenario Mr Survive will eventually get Twitter messages from his family members that they are safe with their positions.

While he is moving his way through debris he meets an emergency team that let him use their satellite communication channel to update all his collected tweets to the centralized server and download incoming tweets to people believed to be in his proximity - given his current position.

After he knows that his family has survived he wants to understand where he could get help. He wants to know where there is fresh water, distribution centers of food, where he can charge his phone, safe shelters and other primary needs. This information is opportunistically exchanged with other people he meets and eventually matched with messages that offers what he is looking for and the position of the facilities. Mr Survive also decided to subscribe to tweets about disaster news and weather forecasts to get a better understanding of the larger situation.

When he met the emergency team he was given a water sensor that could differentiate between fresh, polluted and hazardous water. While wading through water drenched terrain his sensor continuously collects information about the water quality and detects areas of hazards. This information and the precise locations are collected by the smart phone and the positions of hazard water as well as type of pollutions are tagged. When Mr Survive got the alarm about the water he also decided to take a picture of the area that is also attached to the tag. This information is then opportunistically distributed to other people that he meets so that they get a warning if they are approaching the area. A map of the water hazards will also assist emergency teams in their efforts and priorities.

From the disaster scenario described above, we can derive the most important user needs in situation of distress:

- #1: Let someone know that you are safe (and at what given location) and to ensure that close relatives are safe.
- #2: Pairwise communications with relatives.
- #3: Reach an emergency response agency.
- #4: Location of food, water, shelter, charging stations, etc.
- #5: Location of damages, evacuation routes, road closures and hazards.

3. TWITTER IN DISASTER MODE

In the following, we will first list the new requirements imposed to Twitter to fullfill the user needs in disaster and emergency situations where the telecommunication infrastructure is wiped out. We then describe the implementation of our "disaster ready" Twitter client *Twimight*[1]. In particular, we focus on how we have integrated opportunistic communication.

[1]Available for download (requires Android 2.3.3 or newer) under `http://code.google.com/p/twimight/`

3.1 Application Requirements

In the immediate phase after the disaster survivors will try to use Twitter to organize their most urgent needs. Users would expect their Twitter to work as usual. Therefore, adding new features to an opportunistic Twitter should be limited as much as possible to avoid any additional confusion. The inevitable network setup change from an almost real-time centralized Twitter to a delay-tolerant and distributed Twitter should be as seamless as possible. Besides, we strongly believe that the human ability to adapt and self-organize in such situations is unique and that simple communication functions or primitives rather than more complex ones should be provided. Our Twitter in disaster mode should thus be designed accordingly.

Yet, there are multiple technical challenges raised compared to the original server based Twitter, mostly because we are moving from a centralized solution to a delay-tolerant and distributed Twitter. Users will no longer need Twitter to send their tweets to their followers but rather tweets of global value should be spread to everyone. Epidemic spreading is definitely the best solution providing the highest reaching degree and lowest delay but at the cost of a overhead. This overhead will increase with the number of contacts and generated tweets. However, since tweets are only 140 characters, scaling issues are more likely to happen at the human level and its capacity to digest all global tweets. We will discuss scaling issues in Section 4 and 5.

3.2 Prototype Implementation

We have implemented *Twimight* as a disaster ready Twitter client in Java for the Android operating system. Figure 2 gives an overview of the two modes of operations, normal and disaster, which we detail next. *Twimight* accesses user data using OAuth and the Twitter API after the user grants access to *Twimight* by logging in with its Twitter credentials.

Twimight in Normal mode: *Twimight* supports most basic Twitter functionality like showing a timeline (i.e., the most recent tweets of the followed users), sending tweets, re-tweeting and replying to and setting favorites, direct messagges etc. In normal operation, *Twimight* queries Twitter for new tweets every 2 Minutes. The tweets obtained from the queries are cached locally in a SQLite database. Figure 1a shows the timeline in normal operation and the actions a user can take.

Twimight in Disaster mode: *Twimight* has a checkbox in the settings menu to enable the disaster mode as shown in Figure 1b. The normal Twitter client functionalities stay the same but now rely on opportunistic communications. Upon enabling the disaster mode, the tweets of the user are internally assigned the special status of *disaster tweets*. They are stored locally in a separate table for opportunistic spreading and later publication to the Twitter servers whenever connectivity is detected. Disaster tweets are highlighted in red on the user interface (Figure 1c) to mark them as important.

In disaster mode, the device maintains the CPU awake acquiring a wake lock and periodically scans for reachable Bluetooth devices. The scanning interval has been set to 2 minutes $\pm U[0, 20]$ seconds. It represents a tradeoff between energy consumption and delay as a longer scanning interval would miss short connection opportunities. The interval has been randomized in order to avoid interference and decrease the probability of scanning collisions. We choose Bluetooth for opportunistic communication, because in spite its limitations (cumbersome pairing, etc.), it offers an acceptable trade-off between battery lifetime and service provided. Another big benefit is that it can be used without rooting the Android devices. Adding alternative ad hoc communication methods such as WiFi Ad Hoc and WiFi-Opp [] is planned for the future.

Once two phones are close enough to connect to each other, they exchange the new disaster tweets, thereby spreading them epidemically. The data exchange is controlled by a simple communication protocol divived into two phases: the Hello phase and the Data phase. During the former, nodes try to find out what tweets they are respectively lacking. When two nodes meet for the first time, the Hello packet contains the timestamp of the most recent tweet belonging to the other peer. The reason for this is that a node might already have some tweets belonging to the other peer that could have been received indirectly (multi-hop). Sending the timestamp requests the tweets which were published after this time. It can help to decrease the amount of tweets sent by selecting only the missing tweets hence reducing battery draining and connection time. For successive encounters with this peer, the timestamp of Hello messages is no longer used and instead the incremental number of tweets seen in the previous encounters is sent. After receiving the Hello packet the Data phase begins and the nodes start preparing the data to be sent to the other device. According to the information received, only the new tweets will be sent. Those are serialized by an Object Output Stream into packets and sent over the radio channel using the Bluetooth socket. Upon receiving a data packet the devices start processing it in order to extract all the tweets and direct messages and they are added to the local database. In case something unexpected happens, the socket connection is closed by default after 8 seconds.

From Disaster to Normal mode: As soon as connectivity to the Twitter servers is re-established, the disaster tweets sent (or retweeted) by the user are published to the Twitter server.

Modular plugin architecture: Eventually, *Twimight* is flexible and extensible with plugins. One plugin example is described in the next Section and consists of automatically spreading sensor data when in disaster mode. The communication between plugins and Twimight is done by broadcasting intents into the Android system. Twimight has a plugin listener to catch those intents and extracts the information they are carrying which are then processed (e.g., send broadcast tweet with extracted information). To prevent malicious applications from using this mechanisms to send tweets, we have custom defined permissions which have to be granted to the plugin application.

To validate the implementation, we ran first experiments in our office environment, proving that everything works as expected.

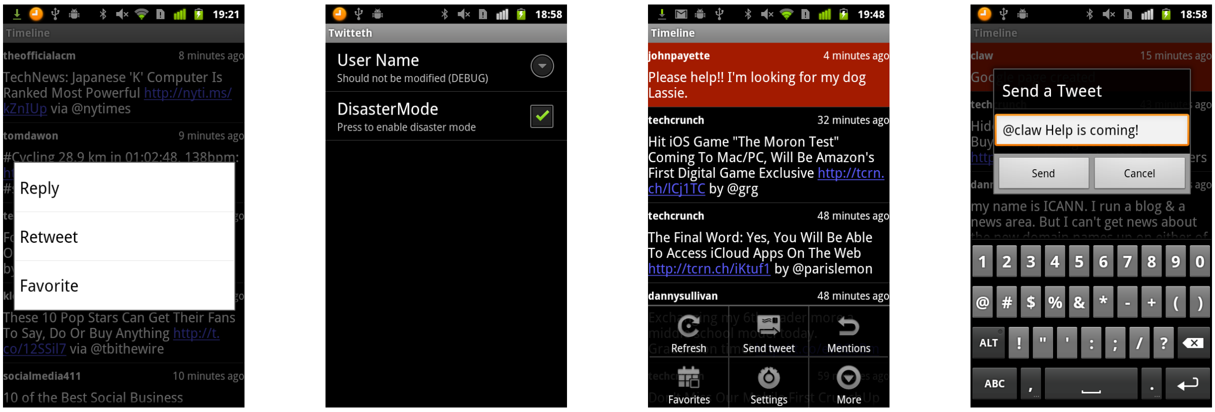
4. SUPPORTING APPLICATIONS

In this section we present two supporting applications for the opportunistic Twitter. The first distributes sensor data with tweets, thus providing additional information about the user's environment. The second increases the efficiency of the opportunistic spreading. Both applications run autonomous without requiring user involvement and do not modify the opportunistic Twitter behaviour.

4.1 Spreading of Sensor Data over Twitter

While the format of tweets allows for a wide variety of messages and images, they are typically user generated, related to social life, interests, etc. We extended Twimight to automatically tweet *sensor data* when in disaster mode. Sensor data can easily be formatted into tweets, do not require user interaction, and provide additional information about a user's environment. This information can be useful by itself to get the context of a person, but it can also be put into a larger context by combining sensor data from other users and crowdsourcing [8, 9].

Inferences about human activity, location and social events has been shown feasible on mobile phones by mobile and participatory sensing applications using microphone, accelerometer, gyroscope,



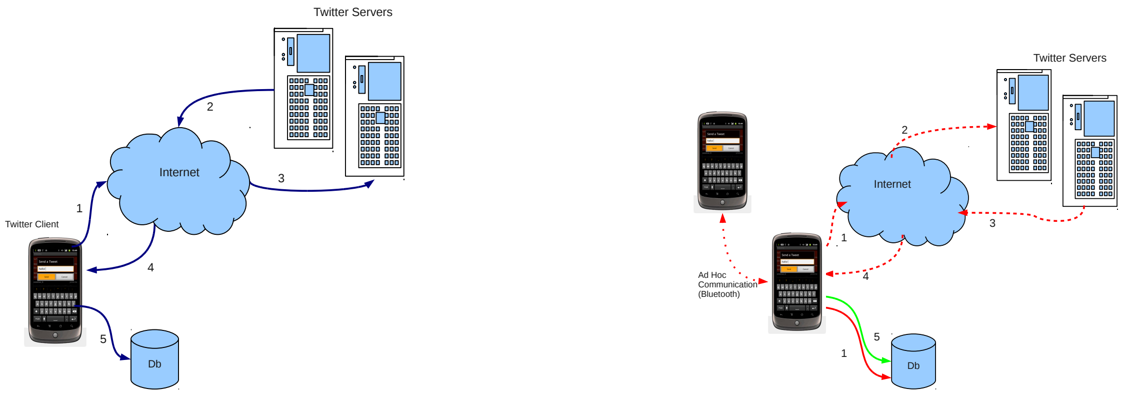
(a) Normal timeline and tweed context menu.

(b) Enabling disaster mode.

(c) Timeline with highlighted disaster tweet.

(d) Sending tweets in disaster mode.

Figure 1: *Twimight* screenshots.



(a) Normal operation.

(b) Disaster mode operation.

Figure 2: Two modes of operation. Continuous blue lines mean normal operations. Dashed red lines refer to attempts in disaster mode, whilst continuous red lines means operations that are performed without problems in disaster mode.

camera, network interfaces, GPS, and other sensors built into or attached to modern mobile phones [10, 11]. In our prototype we provide a stream of aggregated sensor readings from accelerometer and microphone with a moderate update interval of a few minutes combined with intermediate updates on significant changes. The sensor readings are aggregated and translated into activity states {motionless, stationary, going, running} and environment information {silence, voices, car, noisy, fire} respectively (to cope with the asynchronous nature and the message format of Twitter). The purpose to generate a periodic stream of sensor readings is to provide a sign of life. As a proof of concept, the states are decided upon statistical properties of the (low pass filtered) data; updates are triggered on a consistent state change to avoid flip-flopping. In the future we hope to apply more sophisticated data processing, optimize for energy consumption, and to interface with external sensors more specific for disaster situations like carbon dioxide or water quality sensors.

The sensing is implemented as a Twimight plugin in the form of an Android Service that gets started when the disaster mode is enabled. Synchronization with Twimight is done using system-wide broadcast announcements. If location information is available on the phone (e.g., by the means of GPS readings), then the sensing tweets are geo-tagged as supported by the Twitter message format.

4.2 Interest Matching

Twitter's hashtags allow users to follow specific topics of interest. A centralized server does the user matching requests between tweets and hashtags. In an opportunistic setting, there is no centralized server and the matching must be done locally on each smart phone, either manually by the user or assisted by a supporting application.

In straightforward epidemic spreading all smart phones will get all tweets and local searches on tweets can be performed. In reality a replicated distributed database of tweets is created although with inconsistent states and many redundant updates. This is efficient as long as the number of tweets are limited. The problem are not the sizes of the local databases or the search mechanism but the time available for an exchange of tweets when two users meet. This time could be a few seconds when walking people meet each other using Bluetooth for opportunistic contacts. The time will eventually not be long enough to exchange all tweets collected by the each node.

One way to make this more efficient is to first exchange all hashtags and thereafter only exchange the tweets and messages the other party is interested in. This interest selection could be done manually by the user inspecting hashtags and using retweeting, but will not scale up. Tweets that a user is not interested in will also not simply be exchanged and means that tweets that a third party is interested in will not be exchanged when two users meet. We want smart phones to act as mules and carry tweets for other parties. It is, for example, needed in our scenario when a user needs to find family members as soon as possible. Retweeting is in some way a manual forwarding of tweets of interest for third parties, but does not scale up.

To scale further, epidemic spreading of tweets in opportunistic networks must therefore be combined with priority rules on what should be exchanged first at a meeting when there is a limited *bandwidth*time product*. A "last in first out" rule that prioritize fresh data might make sense but does not profit from knowledge about the hashtags. Instead, we believe a data centric approach should be used that match hashtags with user interests. For example, a hashtag for request of immediate help should have priority over other hashtags.

We propose that when two users meet their smart phones first exchange their users interests' in the form of their hashtags but not any tweets until they have been ranked. By successively collecting these user interests and identities from meetings, a node can build a map or graph of other users interests'. By exchanging these graphs with each other the opportunistic nodes gradually build more and more complete graphs of users and their interests in the neighborhood. The idea is similar to link state routing protocols when each node collects the link connectivity seen from other nodes by sending it further and in that way builds a routing table.

Given the knowledge on the level of interests in hashtags and how important they are, the meeting nodes could rank the potential tweets to be exchanged when there is only time for a limited number of exchanges. This will take off the burden from the user to select tweets to exchange but still give her the ability to manually re-tweet and give priority to specific hashtags.

From the interest graph it is also possible to deduce who has met whom and use that information to identify mules (nodes) that are likely to propagate tweets towards the users interested in them. Likewise, the information could be used to rank hashtags for exchange with respect to the level of user interests interpreted from the graph. We have shown in previous work [5], that data spreading through interest groups (such as described above) increases the performance of opportunistic forwarding, including epidemic forwarding.

Also from the graph, more informative decisions could be taken on what tweets to discard when buffers fill up on smart phones. In a recent paper [12] we show that smart data dropping decisions also lead to better utilization. The results show that dropping data based on replication yields the better delivery ratio and the lowest forwarding overhead compared to dropping on the lowest interest.

5. DISCUSSION

Here, we discuss the main technical and non-technical challenges of an opportunistic Twitter used in a disaster and emergency situation. We also review state-of-the-art research from ad-hoc to opportunistic delay-tolerant networking and their suitability. We give some research directions in unexplored fields.

5.1 How to Fully Exploit Twitter Functionalities in Disaster Situations

First, let us consider an interesting Twitter emergency use case. It has been reported that people in the US tweeted for medics instead of dialing 911 in emergency situations. Furthermore, these users were explicitly requesting retweets from people having medics as their followers. The retweets would hence result in a search for medics on the Twitter follower graph, or in other words, the retweet function was used as a *human-based collaborative search and filtering function* on the Twitter social network. Such emergent behavior is also likely to be exhibited in opportunistic Twitter environments. The question now is how this could be supported technically with opportunistic communications while still taking the most out of Twitter's basic functionalities i.e., hashtags and especially retweets in the context of disasters.

On Scaling Network and Human Capacities: To ensure the reliable dissemination of tweets to the greater number, the use of epidemic spreading is mandatory at first. With the epidemic dissemination of tweets, not only do we potentially use a lot of network resources but victims of a disaster can also get overwhelmed with the number of received tweets that are displayed in their timeline. There are hence clear needs to scale such dissemination both at the network and human levels. We detail next how this can be achieved.

The case of Retweets: In a first phase, immediately after the disaster, one can exploit the human expertise and knowledge with retweet featuring a human-triggered dissemination: tweets of people looking for someone or a shelter could be disseminated with a limited range and only retweets would trigger another dissemination further extending the geographic scope of a disseminated tweet. For example, the dissemination of a tweet with information about a shelter location could be limited to, say a 5 km radius from the shelter. Someone coming from outside this 5km dissemination range might know about a shelter which was announced further away. When reading a request for shelters, this person can retweet the shelter announcement. In this way, this person acts both as a tweet mule and provides a human-triggered extension of the dissemination of tweets. Many ad hoc protocols have considered lookups of increasing range to scale but we clearly see that with Twitter the human dimension has completely been overlooked while it can play a key role in scaling lookups.

The case of Hashtags: Later on, when people have organized, information can be filtered by interests and expertise. Medics located at different field hospitals would communicate with each other about drug supplies, available beds, etc. Such communications could still rely on epidemic spreading at first but in order to provide relevant information, the medics' timeline should display only the tweets filtered by hashtags (#drugs, #beds). Assuming that all tweets are spread epidemically and hence stored in the local tweet database, filtering by hashtag would only display the recently matching received tweets. In order to reduce the dissemination overhead, one could resort to interest-based multicast groups. So, in addition to geographic location, the hashtags used in tweets could also be used to further consolidate the network with multicast groups based on hashtag-based matching as proposed in Section 4. Note eventually, that tweets should also be prioritized (to ensure most important tweets are forwarded in case of short contacts) and energy saving strategies are also of fundamental importance in such situations.

The case of Direct Messages: In a similar way, we can leverage the epidemic spreading process to also disseminate the geographic position of people (as described in Section 4). Once enough positions have been disseminated, unicast communications (i.e., Twitter Direct Messages) using geographic routing could be supported in a more scalable way avoiding to flood the all network to reach a destination. Such scenario challenge the usefulness of classical unicast delay-tolerant network routing protocols. However, the ideal routing protocol is probably an hybrid protocol exploiting the respective advantages of geographic forwarding [13], Fish-eye State Routing (FSR) [14], Spray and Wait [15], ProPHET [16] and SimBet [17].

5.2 What are we still missing?

We are still missing mobility traces in disaster situations. This could help better designing protocols but also evaluating empirically such protocols by replaying such traces. Of more general interest, we also wonder what the reaction of users will be to delay. We discuss these two issues next.

On Connectivity and Mobility during Disasters: A major unknown is mobility patterns in such situations. What if mobility is hindered such as during the recent floods, which occurred in News Orleans and Queensland. In the former, a urban environment, one can expect connectivity because of the high population density. In the latter, limited mobility and the low population density might lead to a very sparse topology and infrequent connectivity. In the end, density and mobility clearly drive how information dissemination would perform hence the need for such measurements. Since such information will probably be very hard to collect, protocols must be able to handle a wide variety of situations.

User Reaction to Delays: Until now, we have not considered human reactions to an opportunistic Twitter which by definition would incur delays. We can legitimately ask ourselves what would be the layman reaction to such delays? Would the reaction of a distressed victim be the same? These are fundamental questions that could be answered by experimental psychological studies with voluntary guinea pigs. Those questions could also be partially answered by polling the users of the N4C DTN infrastructure [18] in Lapland.

To summarize, our approach is a top-down approach with our application being designed with emergency and disaster situations in mind. Yet, with the lack of data about such situations and their diversity, it is hard to take any design decisions for they might not be adapted once in situ. This is why we decided to rely on an existing social networking application which provides simple features human beings can grasp and exploit (i.e., tweets, retweets, hashtags). On the technical side, this calls for thinking in terms of system design or how to provide simple features that collectively give rise to emergent properties of self-organization (e.g., world wide web). This also calls for online optimization algorithms which would infer situations and adapt parameters accordingly, the goal being to avoid any fixed parameter in any routing algorithm [19, 20].

6. CONCLUSIONS

In this paper, we proposed an opportunistic Twitter as the ideal emergency kit in situations of disasters or emergencies where the fixed telecommunication infrastructure is down. We have implemented Twimight, a Twitter android application relying on opportunistic communications to spread tweets and sensor data in an epidemic fashion. Putting Twimight in such context and relying on opportunistic communications, we have found many technical and non-technical challenges that needs the community's attention. We plan to experiment our prototype in the field during the 3 days immersion in the Jungle of ExtremeCom 2011 - The Amazon Expedition to collect technical and user feedback. We hope to demonstrate the usefulness of Twimight to all participants.

Acknowledgment

This work was partially funded by the European Commission under the SCAMPI (FP7 – 258414) FIRE Project.

7. REFERENCES

[1] A. Bruns, "Tracking crises on twitter: Analysing #qldfloods and #eqnz." Presented at the Emergency Media and Public Affairs Conference, Canberra, 2011.

[2] R. Bruno, M. Conti, and E. Gregori, "Mesh networks: commodity multihop ad hoc networks," *Communications Magazine, IEEE*, 2005.

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

[4] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *ACM MobiHoc*, 2004.

[5] E. Nordström, P. Gunningberg, and C. Rohner, "A search-based network architecture for content-oriented and opportunistic communication," *Uppsala University Technical Report 2009-003*, 2009, http://code.google.com/p/haggle.

[6] PodNet Project. [Online]. Available: http://www.podnet.ee.ethz.ch/

[7] Delay tolerant networking research group. [Online]. Available: http://www.dtnrg.org

[8] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich, "Mobiscopes for human spaces," *IEEE Pervasive Computing*, April 2007.

[9] Ushahidi platform. [Online]. Available: http://www.ushahidi.com/

[10] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The jigsaw continuous sensing engine for mobile phone applications," in *ACM SenSys*, 2010.

[11] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *ACM MobiSys*, 2009.

[12] F. Bjurefors, P. Gunningberg, C. Rohner, and S. Tavakoli, "Congestion avoidance in a data-centric opportunistic network," in *Proceedings of the 2011 ACM SIGCOMM Workshop on Information-Centric Networking*, 2011.

[13] I. Stojmenovic and X. Lin, "Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, October 2001.

[14] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye state routing in mobile ad hoc networks," in *In ICDCS Workshop on Wireless Networks and Mobile Computing*, 2000, pp. 71–78.

[15] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *ACM WDTN*, 2005.

[16] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE MCCR*, 2003.

[17] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant MANETs," in *ACM MobiHoc*, 2007.

[18] N4c. [Online]. Available: http://www.n4c.eu

[19] J. Karvo and J. Ott, "Time scales and delay-tolerant routing protocols," in *ACM CHANTS*, 2008.

[20] T. Hossmann, T. Spyropoulos, and F. Legendre, "Know thy neighbor: Towards optimal mapping of contacts to social graphs for DTN routing," in *IEEE Infocom*, 2010.