On Building Special-Purpose Social Networks for Emergency Communication

Mark Allman
International Computer Science Institute
Berkeley, CA, USA
mallman@icir.org

ABSTRACT

In this paper we propose a system that will allow people to communicate their status with friends and family when they find themselves caught up in a large disaster (e.g., sending "I'm fine" in the immediate aftermath of an earthquake). Since communication between a disaster zone and the non-affected world is often highly constrained we design the system around lightweight triggers such that people can communicate status with only crude infrastructure (or even sneaker-nets). In this paper we provide the high level system design, discuss the security aspects of the system and study the overall feasibility of a purpose-built social networking system for communication during an emergency.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design

General Terms

Design

Keywords

Emergency communication, Social networks

1. INTRODUCTION

Emergency situations—ranging from natural, such as earthquakes, to man made, such as terrorist attacks—come with their own unique set of communications requirements:

- First responders (police, fire fighters, medical personnel) have obvious and acute need to coordinate activities and collect status reports about the unfolding emergency situation.
- Individuals have a need to report various situations to authorities (e.g., that someone is trapped in a house that is surrounded by flood water).
- Authorities have a need to inform the public (e.g., evacuation orders).
- While not nearly as important, normal people inside a disaster area have a need in the immediate aftermath of a disaster to communicate with friends and loved ones outside the disaster area to report their own individual status ("I'm fine", etc.).
- Finally, there are longer term communications needs that are not as urgent as those listed above but may be difficult in the after-math of a disaster (e.g., coordinating water and food distribution, communicating with the public about when it is safe to return, etc.).

These tasks are complicated due to both the volume of communication that is being attempted within a fairly modest geographic area and also because the usual communication infrastructure is often compromised by the disaster.

To date most of the work on emergency communication has come in the form of (*i*) systems aimed at helping first responders to better communicate and coordinate [11, 8, 6], (*ii*) systems that allow for quick instantiation of fairly ad-hoc networks to aid crucial recovery tasks (in hours to a couple of days) [6] and (*iii*) systems that allow emergency information to be transmitted when communicating over the Internet as it is over the PSTN [2, 10]. In this paper we address the need for individuals in disaster zones to communicate their status with people outside disaster zones. Such communication may seem unimportant at first blush, but being able to let someone know their status in a lightweight fashion can ease the overall stress on both the surviving communication infrastructure and the people caught up in the emergency—which is not a minor issue [6].

Our goal is to enable normal individuals caught up in *large-scale emergency situations* to tap into a pre-arranged social network to give *small bits* of status to their friends and family. We offer a simple architecture for a purpose-built social network for this application rather than tapping into one of the established social networking sites due to the particular challenges presented by communicating in a time of disaster with little infrastructure (see § 5 for more discussion on leveraging existing social networks).

Our high-level vision is that as a preparatory step individuals register with the In Case of Emergency (ICE) system and define a network of contacts to be notified if the individual finds themselves in a disaster.[1] The system assigns each person an eight character identifier that can be fed back to the system at any point (with a short message) and the person's defined social network will be sent the given message. This provides a way for most of the smarts of the system to be kept distributed and therefore outside the disaster area while only requiring a very small bit of information to be

[1]Much like the many campus warning systems that are now in use whereby individuals ask to receive emails or text messages when an on-campus emergency occurs.

```
I:abcd1234
O:Mark Allman <mallman@icir.org>
E:alice@example1.com
E:bob@example2.org
E:charlie@example3.net
S:+11234567890
V:+11234567891
A:charlie_AIM_ID
```

Figure 1: Example ICE record.

conveyed from the disaster area itself. While to ultimately trigger notifications the identifiers have to find their way to the Internet these small IDs are amenable to being written down on paper logs, shown on television crawls, sent via low bandwidth channels, sent via Morse code over ham radios, etc. to propagate out of the disaster area. Further, such IDs can also be written down and kept in a wallet or stored in a cell phone and used if a person is found hurt or deceased and cannot take any action to invoke their notification network themselves.

2. ARCHITECTURE

In this section we discuss the ICE notification system we have designed. We note that a number of the decisions are motivated by security considerations discussed in § 3.

2.1 Identifiers

As noted in § 1 individuals register with the ICE system and are assigned a random eight character ICE identifier (IID). Compact identifiers make them amenable to transmitting across very low speed networks—including carrying such identifiers physically to get to an Internet connected location. In addition, we also desire identifiers that are difficult to tie to a particular person and/or guess (to fabricate notifications) as discussed in more detail in § 3. Finally, while a more meaningful identifier might make these a bit less cumbersome for users to deal with we do not believe IIDs are particularly burdensome for people. In particular, we note that (*i*) IIDs are not used often and therefore being slightly cumbersome when pressed into service is likely not a big problem, (*ii*) people have proven capable of using similar identifiers (e.g., phone numbers, social security numbers, student ID numbers, etc.) in various settings and therefore the makeup of the IID seems reasonable and (*iii*) IIDs could be stored in a person's phone, wallet, purse, etc. so they would not be required to memorize the identifier.

2.2 Records

Next we turn to records in the ICE system. Records encode a user's social graph for the purposes of emergency notifications. We do not envision some sort of new communication method or application for ICE notifications. Rather, we envision re-using the standard ways people currently communicate, such as email, voice mail, SMS messages, instant messages, OSNs, etc. Records, therefore, associate an IID with a set of notification targets, T_n. When a notification comes into the system for IID n the message will be transmitted to each target in T_n. Each target has a method (e.g., email) and an address (e.g., a standard email address).

An example record is given in figure 1. The record starts by giving the IID (first line) and the owner's name and email address (second line). The user's T_n is then defined by the remaining six lines: the first three lines are addresses to notify via standard email (method "E"), the fourth line denotes a phone number to be sent an SMS message (method "S"), the fifth line provides a phone number for an automated voice call (method "V") and the last line indicates an AIM buddy name to notify via instant message (method "A"). We stress the above example is illustrative and not exhaustive in that we wish to encompass myriad ways in which people generally communicate. For instance, another method that could be added would be to send a notification via a traditional OSN. For instance, the ICE system could post a message to a user's Facebook wall. Also note, the example record is cryptic and could in principle be encoded in something more standard like XML. However, we are sensitive to record size as the system scales to millions or billions of people. See § 4 for a discussion about the aggregate storage requirements of the system.

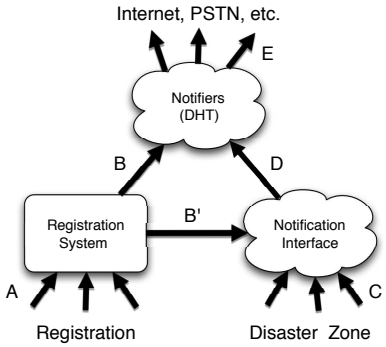

Figure 2: ICE system overview.

Also note that the example shows only a small social network of people that will receive notifications. This is not intended to be a toy example, but rather we believe that allowing users to encode only a small social network is the right approach to allow the system to scale. As discussed in § 4 when the system is pressed into use the aggregate load could be large. Our vision is to push notifications out to a few people outside the disaster area and then lean on those people to propagate the information within wider social networks. Therefore, it is useful to encourage people to carefully construct their ICE notification network. For instance, a network that contacts an individual's parents and siblings may ultimately not allow notifications to propagate into wider social networks. Including a mix of people from one's family, friends, work, school, etc. will allow for more timely notifications to the person's broader social network.

Finally, we note that our vision is for users to update their records periodically (or at least verify them) and to allow the system to purge stale records. For instance, the ICE system may ask a user to verify their records if they have not been changed in a year or two.

2.3 System Overview

Figure 2 shows the overall layout of the ICE system. We first discuss each component of the system and then step through the communication between the various components.

Registration System: This component of the system is where users register and setup their social networks using a standard web-based front-end. This component can be readily centralized as it is not involved with sending notifications during times of emergency, but only with configuring the system before-hand.

Notification Interface: This component accepts IIDs and messages from the disaster area and passes them along to the actual notifiers (see next paragraph). This component consists of a modest number (≈20–30) of distributed hosts that are all listed under one entry in the DNS. These hosts must be distributed and the data they hold fully replicated such that many of them will be outside of any particular disaster zone and therefore together form a reliable system. The interface between the notification interface and the reporters in the disaster zone must be lightweight.[2]

Notifiers: This component consists of a relatively large number of servers (tens to hundreds; see § 4) that form a distributed hash table (DHT) to both hold the database of records and transmit the notifications. We use a DHT to both spread the load across many servers and leverage redundancy in doing so as DHT nodes may ultimately find themselves in the disaster zone. Finally, note, that depending on who runs the ICE system—see § 4.5—the DHT nodes may be distributed among a large number of institutions and therefore we develop a model where we do not place full trust in terms of data visibility in the DHT nodes.

The system works as follows (with the steps corresponding to the labels on figure 2):

A: Users register with the ICE system via a web-based interface and setup both who is in their notification network and how these people should be notified. The system assigns each user a random IID.

B and B′: After a user registers or updates their information the registration server encrypts their record with a new symmetric key, K, and transmits the record to the notifier DHT for storage (step B). In addition, the (IID,K) pair is sent to all notification interfaces (step B′). This encryption setup is used to hide user data until the time it is required (see below) from various hosts in the system—which may be donated by disparate organizations (see § 4.5). Finally, note that operations B and B′ could be periodic batch jobs for efficiency.

C: During a disaster people send their IIDs to the notification interfaces. Since all the notification interface servers have the same information (i.e., the (IID,K) pairs) we can load balance across the servers using round-robin DNS or an anycast address mechanism. Communication with the notification interfaces can be as simple as sending a single UDP packet. However, it is likely useful to develop a range of mechanisms to reach the notification interfaces to accommodate various network policies that might be in place on whatever network a user happens to connect to during a disaster (i.e., firewalling or NAT issues). We leave the actual engineering details—although important—to future work as this paper is meant to contain only a high-level sketch of the ICE system.

D: After the notification interface gets a notification request it passes that request along with the symmetric key required to decrypt the given IID's record to the notifiers DHT which then routes the notification request to a node that holds the given record.

E: Finally, a notifier with the given record decrypts the record and sends out the given message to the network of contacts encoded in the record.

The ICE system could be simplified by removing the notification interface nodes and simply having notifications be sent to the DHT nodes directly. However, as discussed in § 4 large disasters will potentially require at least dozens of notifier nodes to transmit notifications efficiently. Therefore, the ultimate system may rely on disparate organizations to donate notifiers to the pool (see § 4.5). In such a case, users may not wish to have their private data readable by these myriad organizations until their data is needed. Therefore, while the system relies on the notifiers to do their job when called upon the system sketched above does not place a large amount of trust in those systems in terms of data handling.

Note, a notifier can send a notification through the notification interface to coax the notification interface to divulge the key material. However, the rough system we have put in place prevents wholesale harvesting of information. Further, there is nothing in an IID itself that suggests who the owner is and therefore targeting individuals with bogus notifications would be difficult. Finally, the notification interfaces could likely detect large-scale record decryption activities.

2.4 Testing

One of the key issues with infrequently used systems such as ICE is ensuring they are operational when they are needed. Campus warning systems, fire alarms, tornado alarms, etc. all share this problem. We envision ICE could conduct the equivalent of periodic "fire drills" to ensure the overall system is operating correctly. This could be done by choosing a small number of real IIDs to which test notifications are transmitted. Or, it could involve simply fabricating a number of IIDs that are propagated to the various components as normal and then notified via the standard methods. Either of these approaches would illuminate problems in the overall system.

3. SECURITY

Our application area dictates a fundamental tension between needing ease of use of a lightweight triggering mechanism and securing the mechanism against mis-use. Precisely because an emergency environment places constraints on the

[2]Note, we assume the notifications are coming from the disaster area in designing the system. However, this is not strictly necessary. For instance, an IID may be transmitted out of the disaster area via ham radio and therefore the report coming into ICE is coming from some well-connected host in an unaffected area.

communication infrastructure we design our system to be lightweight and not require interactive and secure mechanisms to trigger notifications. If such mechanisms are available then users are likely better served sending emails or posting information on an established social network site. Therefore, to protect the system and the users we resort to both security through obscurity and implementing policies that follow the expected use case of the system. This results in a *roughly secure* system, but that does not mean individual operations will always be legitimate. However, our position is that without additional burdens on either the users or the requisite communication infrastructure we cannot expect much more than rough security.

The first concern with our system is in terms of forgery, which can manifest in two ways: (*i*) forging a registration and (*ii*) forging a notification. The former is difficult to prevent in absolute terms. The registration system can employ standard techniques that attempt to mitigate the impact of automated registrations (CAPTCHAs [13], confirmation emails, etc.) This leaves the problem of how a recipient can know their acquaintance actually created the IID that tripped a notification as opposed to someone forging the IID record. One way to accomplish this is to allow the creator to augment a record with some information that is not generally known and will increase the trust the recipients place in the notification.[3] This is similar to security questions web sites use to help vet users upon login. While this mitigates the problem of arbitrary registration forging it does nothing to prevent either targeted registration forgery or malicious triggering of notifications.

To prevent spurious triggering of notifications we lean on two imperfect notions: (*i*) IIDs are random, obscure and the IID-space will be sparsely populated and (*ii*) outside context. That is, we rely on the fact that it will first of all be difficult to guess *any* valid IID given a space of over 2.8 trillion and a world population of 7 billion. Further, simple guessing will be readily detected in a fashion similar to the way scan detectors identify scanners who are blindly guessing at hosts and ports to probe [7]. Guessing an individual who is actually plausibly involved in a given disaster is even more difficult, which is where the second part of our mitigation comes into play. The notifications will be sent to the owner of the given IID, as well as their pre-configured social network. Therefore, forging a notification that, for instance, claims someone has been found in a hospital (or worse) in the aftermath of an earthquake can be quickly corrected by the owner of the IID or some member of their social network who, say, happened to just see the person thousands of miles from the disaster area.

A closely related issue is that of laundering messages through the ICE system to mask a sender's identity. This would work by registering an IID and configuring it with a set of people to be contacted when a notification arrives. The message is sent to the IID through the notification interface and passed along to the list of contacts. This obscures the true source of the message itself and amplifies the message sent by the attacker. We mitigate this by first trying to prevent automated registrations (as sketched above). Further, by constraining the size of the encoded social network as discussed in § 2.2 the incentive for laundering is modest. For instance, a spammer would be able to reach only a small number of people with a single laundered message.

Another general attack on the system is a resource usage attack. For instance, registering massive lists of people to notify or trying to send fine-grained updates every few minutes from a disaster area. To mitigate these issues we leverage the intended use case of the ICE system that suggests some common sense policy decisions:

Limited Social Network: The number of people who can be notified by a single IID can be limited. This allows the resources to be used broadly across IIDs. Further, this relies on additional social networks (network-based, word-of-mouth, etc.) to carry the notifications widely.[4]

Breadth First: The notifiers should likewise optimize for breadth first. In other words, given a list of notifications each should be sent to one of the contacts before any notification is sent to a second contact.

Best Effort: If a notifier is in the midst of sending notifications on behalf of some IID and a new notification for that IID arrives the old notification should be aborted in favor of the more recent notice. In other words, notifications are not guaranteed to be sent, but should be considered best effort. And, in times of resource contention some notifications will be lost. (That said, simply discarding the only notification for some IID should be highly unlikely.)

Limited Use: An IID should only be invoked in an emergency and policies can be developed to ensure this is the case—or the IID disabled. Emergencies happen only every once in a while and therefore observing an IID being used every day would constitute a violation of the spirit of the system or worse using the system to launder messages. The ICE system, for instance, could treat IIDs as one-emergency identifiers and deactivate some amount of time after their first use, requiring the owner to login to the registration system to obtain a new IID or re-activate the current IID.

Within the framework sketched in the last section the notification interfaces present a reasonable "firewall" of sorts for notifications. IID usage can be tracked and regulated at this location. Further, there will be few enough of these nodes to make sharing log data tractable—especially given that in the absence of an emergency we expect the systems to be used very little.

4. PRACTICALITY

We now turn our attention to the practicality of the ICE system. Our focus is on running a single system for the entire

[3]In practice, multiple such statements may be needed since one may not well cover disparate acquaintances in one's social network.

[4]An alternate design could allow for larger social networks that also include priorities such that notifying a large group of people for one IID does not impede notifying a small group for another IID.

world, however, it is possible to instantiate smaller systems with narrower focus (e.g., a country). While prototyping and building the system would allow for a rigorous understanding of the requirements we here undertake an initial feasibility study to assess whether the system will plausibly scale to the size possibly required. Therefore, we use some conservative information to drive our understanding. With that in mind we offer the following bounding parameters before turning to several issues of practical import:

Population: At the time of his writing the world's population is nearing 7 billion. For our purposes we will assume each person in the world will register with the system. This is clearly a gross exaggeration on a number of axes. However, it is also likely the case that some people will register more than once (e.g., because of a forgotten IID).

Disaster Size: The largest disaster in terms of plausibly directly effected people with an acute need to communicate status that we can conjure are the terrorist attacks of September 11, 2001 on New York City and Washington D.C.[5,6] The U.S. Census Bureau reported that in 2000 there were 17.8 and 3.9 million people living in the New York City and Washington D.C. "urban areas", respectively [12]. We, therefore, use 21.7 million as an upper bound on the number of people who might plausibly be trying to send notifications during an emergency. Even though there were many visitors in both cities that day we also feel that all of the residents of the urban area were not in the directly impacted areas and therefore did not have either acute notification needs or experience crippled communication infrastructure.

Timeliness: Finally, to be useful ICE notifications need to be timely. Therefore, we place two constraints on our analysis. First, all notification requests arrive instantaneously (an unrealistic worst case). Second, we wish to push out at least one notification per IID in the first hour after a disaster. This is an arbitrary time period, but provides a touchstone that allows us to gauge the system requirements.

4.1 Storage Requirements

As a basis for understanding ICE's storage requirements we start with the example record in figure 1 and replaced the email addresses with the longest three addresses in the author's address book. In addition, we added a sentence that represents a fact likely known by only the author's acquaintances (per the discussion in § 3). The record ended up being 238 bytes long. We then encrypted the record using *gpg* [1] with a symmetric key and the resulting record was 251 bytes. Records will require additional overhead (time-to-live, database indices, etc.) and therefore we conservatively assume each record consumes 1 KB. The following are rough estimates of storage requirements on the three components of the system:

Registration Server: Assuming a record for every person on the planet at the registration system requires 6.5 TB of storage. As a data point, a Drobo-S 7.5 TB storage system is listed at $1375 at the time of this writing.[7]

Notification Interface: Each notification interface requires a copy of the database of IIDs-to-symmetric key mappings. Assuming 1024 bit keys[8] this requires 136 bytes per pair or 887 GB for 7 billion pairs. Internal 1 TB SATA disks are available for less than $100.[9] For robustness we envision each notification interface server having two mirrored disks. Per § 2.3 we envision 20–30 such servers for redundancy. Therefore, a high-end estimate for this component of the ICE system is $6000.

Notifiers: Finally, we need to replicate and keep redundant copies of all encrypted records in the DHT created by the notifiers. The cost depends on the number of replicas. Just for storage cost estimates assume four replicas of each record. This requires 26 TB of storage. For $2600 we could outfit each of 26 DHT nodes with a 1 TB disk for this task.

Overall the system requires roughly 60 TB of storage capacity distributed across dozens of hosts, costing less than $10K. In terms of storage, we find the system to be feasible even if additional capacity or redundancy is desired.

4.2 Notification Transmission

While the system's storage requirements are overall modest a larger challenge is transmitting an immense number of notifications when an emergency hits. For our initial discussion we assume all notifications are sent via email. Recall that our goal is to send at least one notification per IID within one hour of the outset of an emergency. Using our upper-bound emergency size this requires that 21.7 million emails be sent within an hour—or, 6,207 emails/second. In discussing email origination rates with colleagues it seems that 100 emails/second should be readily achievable on reasonable server-class hardware (especially if the software configuration is optimized for such sending). In particular, Paxson [9] reports 35 instances of a single host opening at least 100 outbound SMTP connections in a single second from a connection log taken at the Lawrence Berkeley National Laboratory (LBNL) in mid-February 2010. These instances

[5]We concentrate on the number of people who may need to communicate in our analysis. The attacks of September 11^{th} are of course dwarfed by other disasters when using alternate metrics (e.g., loss of life, property damage, etc.).

[6]Note, there are larger scale "events" that we do not necessarily consider to be "emergencies" in that they do not trigger immediate need to contact one's social network. For instance, [4] lists large scale blackouts that have impacted upwards of 100 million people.

[7]See http://www.drobostore.com/.

[8]This key size seems sufficient to prevent casual snooping—which is all we can hope to prevent given that the notifiers can coax the system into divulging any IID's key (per discussion in § 2.3). Further, while a larger key size would cost more disk space the additional cost is not infeasible (e.g., for 2048 bit keys).

[9]See http://amazon.com/.

span 5 hosts and the peak observed rate was 147 connections/second. Assuming 100 emails/second a set of 63 notifiers could meet the requirement to send 21.7 million notifications within an hour.

To assess the required network capacity of sending 21.7 million email notifications within an hour we first note that messages from the emergency area are meant to be short and we therefore limit them to 140 bytes.[10] We also note that over the first three weeks in February 2010 the average and median header size on the author's outgoing mail was just over 500 bytes. Therefore, using 1 KB as a rough estimate of the payload size seems reasonable. That leads to a requirement for the system to send just under 6 MB/second. If, per the above analysis, we assume this is spread across 63 hosts then the rate is just under 100 KB/second. Even if additional overhead increases the data transmission requirements by a factor of two or three the resulting rates are not unreasonable for well-connected hosts.

We do not have any solid notions at present about how efficiently non-email notifications could be transmitted. In some cases we can guess that the process would be arduous (e.g., automated voice calls). In other cases it seems that the process would be at least as quick as email. For instance, we may be able to develop a mechanism for sending a large number of instant messages efficiently to some centralized system (e.g., AIM). Therefore, while we use email as the basis of our feasibility calculations the required resources may be more or less. In addition, the system may prioritize notifications based on the transmission efficiency of the method. That is, the notifiers might prefer to send all email notifications before engaging in automated voice calls. Finally, while our vision is that myriad notification techniques are useful and the resulting framework accommodates various notification methods it may be necessary to scope the actual methods used to only those that can be efficiently delivered from a wide variety of Internet-connected nodes.

4.3 Database Updating

We next consider the database updating load on the registration system. Assuming 7 billion records and each being updated once per year, per the discussion about keeping records current in § 2.2, an average of 19.1 million records will be updated per day (or 222 updates/second). While this is not an overall onerous load given some of the large web applications in use on the Internet it is likely too much for a single server to handle. Therefore, the registration system should be implemented as some form of server farm that an be expanded as necessary to handle the update load.

4.4 Aggregate Cost

In terms of aggregate hardware cost we assume $2K per server.[11] Per the analysis in § 4.2 we need 63 notifier hosts which we will outfit with 1 TB disks.[12] Further, per the storage and redundancy requirements for the notification interface servers sketched in § 2.3 we assume 30 such servers, each with two (RAIDed) 1 TB disks. The cost of all 93 machines is just under $200K. Therefore, adding a modest server farm (with storage) to perform the registration system duties the entire price tag for the hardware would be less than $250K.

The servers that constitute the ICE system all need network services and therefore hosting costs are also a factor. Sprocket Networks[13] offers co-location services starting at $75/month per server. Assuming that rate it would cost $84K/year to host the notification interfaces and notifiers. The registration system would likely need to be a farm and therefore would cost more than a simple server.

Finally, note that the ICE system will have an ongoing administration requirement that must be worked into the costs. Most of the servers will fall into two classes (notifiers and notification interfaces) and within those classes the servers will be highly similar. This eases the management headache considerably making it a task that a small team of people spending part of their time could likely undertake.

4.5 Who?

Given the above analysis we believe that the ICE system is technically feasible. However, this still leaves two questions: (*i*) who will pay the costs? and (*ii*) how will network connectivity be handled?. We believe there are several models that could be feasible.

An organization such as the International Red Cross or the United Nations could run the system with dues or donations. Similarly, a new and independent organization could take on the task, recouping the cost by charging a modest fee for IIDs or charging for advertising on the registration server. This could work similar to the non-profit MedicAlert Foundation [5] which provides people with chronic medical conditions a necklace or bracelet with an identification number that first responders can use to get crucial medical information via a phone call when the person cannot readily give the information themselves.

Alternatively, a number of large companies already possess the infrastructure to provide user services (e.g., email, calendaring, photo storage, etc.). In that context the ICE system could likely readily be absorbed into the organization's infrastructure. We do note that it is desirable for the notifications to be generic and be available outside of some closed system. I.e., the people being notified should not have to belong to some particular OSN to be notified.

A final model is a more distributed and grassroots model. In this case, some entity such as the Red Cross would run the centralized registration server, but the remainder of the system would be donated by and housed at disparate organizations to distribute the cost (e.g., at hospitals, universities, benevolent companies, etc.). This could work well because the cost for any one component of the system is quite modest. One concern of such an ad-hoc set of hosts composing the system is that of reliability. However, in this grassroots model the system could be built with much more redundancy

[10] This is also consistent with possibly delivering the messages via SMS.

[11] Reasonable server-class machine from http://www.dell.com/.

[12] Note to handle the storage load and redundancy targets sketched in § 4.1 1 TB disks cover more than necessary. However, dropping to 500 GB disks is a modest cost savings and so we use the higher price.

[13] http://www.sprocketnetworks.com/

than sketched above since the costs of that redundancy are widely distributed. This in turn provides reliability out of an ad-hoc set of hosts.

Note, the model chosen will have some impact on the system design. E.g., if one organization runs the system we could forego keeping the records encrypted at the notifiers.

5. WHY NOT AN EXISTING OSN?

A natural question to pose is: why build a specialized OSN for notification during an emergency instead of leaning on one of the many existing OSNs? In particular, Twitter seems attractive given its lightweight operation and focus on small messages. If the communication infrastructure will support accessing an existing social network that may well be a better avenue for sharing one's status. However, in times of emergency we assume the communications infrastructure will at least be heavily constrained if not unavailable. In this case there are several problems with relying on standard social networking systems. First, as noted in § 2.3 ICE messages can be sent in as little as one UDP packet, whereas existing social networks require more interaction to authenticate and send a message. Second, triggering a notification on behalf of a user is either not possible or requires the person to turn over their credentials to some third-party. Third, identifiers in existing networks are well-known and therefore simply advertising an ID and a message without authenticating to the system is both easy to abuse for fraudulent notifications and can impact privacy. Finally, relying on an existing system will likely scope the notifications to only that system and not to people whom one might want to notify but who do not use that particular system. Therefore, we believe a purpose-built system best meets the requirements of our application.

6. PROTOTYPE

We have built a small prototype ICE system. While it does not have all the particulars of the envisioned system it illustrates the concepts. The tools are available at [3]. The tools allow a user to register and obtain an IID and then add and remove email notification addresses to the IID's record.[14] In addition, we provide a tool to send a small notification message to a given set of IIDs. At present all application-layer communication is conducted using XML-RPC over HTTPS. The tools are crude, but illustrative.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have discussed the design of a system that allows ordinary people to send short status messages to their social network when they find themselves caught up in a disaster. By defining a standard way to send such notifications we believe that conveying information in such crucial situations will be more systematic, orderly and ultimately more effective. Further, by devising such a system in advance of emergencies we hope to reduce the stress on traditional forms of communication which may be dramatically degraded during an emergency and also be vital for coordinating first responders. We have conducted a feasibility study of the system and find that on technical grounds the framework is sound. Finally, we have developed a small illustrative prototype. Our hopes with this paper are to spur discussion of (*i*) communication by ordinary people during times of emergency and (*ii*) purpose-built social networking applications that have unique requirements that are not readily handled by current social networks.

[14]Note: The service right now only provides email notification as an initial prototype.

Acknowledgments

A Facebook post by Christine Allman initiated the thinking on the system described in this paper. Randy Bush, Philip Hazel and Christian Kreibich engaged in discussions about mail server performance. Vern Paxson crunched LBNL logs for the SMTP server data given in § 4.2. This work was support in part by the National Science Foundation under grants CNS-0831780 and CNS-0831535.

8. REFERENCES

[1] GNU Privacy Guard. http://gnupg.org/.

[2] IETF Emergency Context Resolution with Internet Technologies (ECRIT) Working Group Charter. http://www.ietf.org/dyn/wg/charter/ecrit-charter.

[3] In Case of Emergency (ICE) System Web Portal. http://ice.icir.org/.

[4] List of Power Outages. http://en.wikipedia.org/wiki/List_of_power_outages.

[5] MedicAltert Foundation. http://www.medicalert.org/.

[6] A. Botterell. Tools of Resilience: Requirements for Emergency Networks, Nov. 2007. Presentation at NSF FIND PI Meeting.

[7] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *IEEE Symp. on Security and Privacy*, 2004.

[8] D. Malan, T. Fulford-Jones, M. Welsh, , and S. Moulton. CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care. In *MobiSys 2004 Workshop on Applications of Mobile Embedded Systems*, Apr. 2004.

[9] V. Paxson. Personal Communication, Feb. 2010.

[10] H. Schulzrinne and R. Marshall. Requirements for Emergency Context Resolution with Internet Technologies, Jan. 2008. RFC 5012.

[11] United States Dept. of Homeland Security. Fact Sheet: Achieving First Responder Communications Interoperability: a Local, State, and Federal Partnership, 2004. http://www.dhs.gov/xnews/releases/press_release_0529.shtm.

[12] U.S. Census Bureau. Population of Urban Areas, 2000. http://www.census.gov/geo/www/ua/ua2k.txt.

[13] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In *Eurocrypt*, 2003.