Delay Tolerant Disaster Communication with the One Laptop Per Child XO Laptop

Daniel Iland
University of California, Santa Barbara
iland@cs.ucsb.edu

Don Voita
University of California, Santa Barbara
don@cs.ucsb.edu

Elizabeth Belding
University of California, Santa Barbara
ebelding@cs.ucsb.edu

ABSTRACT

In this paper, we describe the design, implementation, and evaluation of a mesh network based messaging application for the One Laptop Per Child XO laptop. We outline the creation of an easy-to-use OLPC Activity that exchanges Ushahidi-style messages with nearby OLPC users through the Internet or a mesh network. Our contributions are to implement an epidemic messaging scheme on mesh networks of OLPC XO laptops, to extend the Ushahidi web application to efficiently exchange messages with nodes in mesh networks, and to allow the Ushahidi server to distribute *cures*, notifications of message delivery, for each received message. Testing and analysis revealed substantial overhead is introduced by the OLPC's use of Telepathy Salut for activity sharing.

Keywords

Ushahidi, OLPC, Mesh Networking, Peer-to-Peer, Epidemic Routing, Telepathy Salut, Disaster Communication, Situational Awareness, Delay Tolerant Networking, Information Sharing

INTRODUCTION AND MOTIVATION

In the hours and days following a disaster, victims and first responders must plan their actions, using whatever information is available. Helping victims and first responders share information with each other can improve their outcomes, by enabling them to make informed decisions and assist each other (Palen, Hiltz, and Liu, 2007). Sharing recent, relevant information can improve aid distribution, search and rescue efforts, damage assessment, personal safety, and other humanitarian efforts.

One of the most popular web applications used for sharing information following a disaster is a crisis mapping framework, Ushahidi. Ushahidi is an open source application for receiving and displaying geographically tagged information. Since 2009, more than 20,000 Ushahidi deployments have been created, including Ushahidi instances for major disasters such as the flooding in Pakistan in 2010, the earthquake near Christchurch, New Zealand in 2011, and the 2011 tsunami in Japan. Ushahidi accepts submissions primarily using text messaging (SMS), email, or Twitter. However, uploading or receiving information from Ushahidi requires Internet access or cellular connectivity. If this infrastructure was present before a disaster, it may be rendered useless by loss of power, loss of network connectivity, or physical damage during a disaster. Increased use of mobile networks during a crisis will likely exceed the capacity of remaining equipment, causing what cellular service remains to be overloaded and unreliable (Hughes, 2008).

There are a number of existing technologies that can be leveraged to facilitate communication post-disaster, particularly when the previously existing infrastructure is overloaded, damaged, or otherwise unavailable. Mesh networks facilitate direct peer to peer communication between Wi-Fi enabled devices. Since messages can spread from person to person, a functioning network can operate without fixed infrastructure. While Internet access is unavailable, delay tolerant messaging techniques, such as store and forward of messages, can be used to spread information through the affected area. As individuals carrying mobile devices move around the area, information is spread opportunistically from device to device, exploiting the movement of device owners (Chaintreau et al., 2006). This flooding-based spread of messages maximizes the probability of messages reaching the Ushahidi server, as a single mobile device that obtains an Internet connection can upload messages

originating from a number of users (Vahdat, 2000).

Due to their incorporation of Wi-Fi mesh networking support, the One Laptop Per Child XO laptops can serve a valuable role in the early stages of a disaster recovery effort. In this paper, we detail the design, implementation, and testing of an Ushahidi-based delay tolerant disaster communication system for OLPCs. We demonstrate how an Ushahidi server can be modified to communicate efficiently with OLPCs in a mesh network, minimizing unnecessary retransmissions between OLPCs and between OLPCs and the Ushahidi server.

DESIGN AND IMPLEMENTATION

In order to work within the existing ecosystem of applications on OLPCs, and to utilize the provided OLPC mechanisms for collaborative applications, we developed our messaging application as a Sugar activity. A Sugar activity is a bundle of Python code and associated libraries, which can take advantage of the services provided by OLPC's Sugar interface. The Sugar interface was designed to be easy to use, and allows OLPC users to easily create or join an 802.11s Wi-Fi mesh network. Activities can be shared amongst users on mesh networks, allowing for two-click download and installation of the activity. Mesh networks and shared activities are automatically detected and displayed in the OLPC's *Neighborhood view*, even if a user does not have the activity installed. Therefore users can join and use the disaster messaging application without advance preparation or software installation.

Sharing between OLPCs is primarily accomplished by two services, an interprocess communication system called D-Bus and a real-time communication framework called Telepathy. Telepathy allows applications running on different machines to communicate via an abstraction called *tubes*. *Tubes* are used to pass text and data, and can be implemented via a number of backend Connection Managers or protocols. OLPC XOs use the Telepathy Salut protocol when connected to a mesh network. On the network layer, Telepathy Salut uses multicast DNS (mDNS), and link local Extensible Messaging and Presence Protocol (XMPP) for device and service discovery. The XMPP protocol enables server-less messaging between clients via mDNS. Multi-user messaging, and therefore activity collaboration on the XO, is accomplished via an extension of XMPP called the Clique protocol. We will discuss the Clique protocol further in our Testing section, as Clique packets constitute a surprisingly large portion of traffic in our experiments.

Implementation of Message Sharing

We utilize an epidemic messaging scheme for sharing application data. When an OLPC joins a mesh network, it transmits messages stored in its persistent storage to nearby OLPCs. Each message is a tuple containing a title, description, category, location, and time. These messages will be retransmitted by OLPCs on the mesh until they have been stored by all reachable OLPCs. The 'new' OLPC will then receive all messages stored by other OLPCs that it did not already possess. To implement this data synchronization, we utilized a data structure called a CausalDict, provided by the GroupThink library. A CausalDict is a Python dict (or hash table) that is automatically shared between all XOs that have joined a shared activity. We use two CausalDict objects, one containing all messages and the other containing all *cures*.

Reducing flooding with cures

One drawback of epidemic routing is its high overhead and resource consumption, due to the large amount of network traffic generated by replicating and retransmitting messages (Song and Kotz, 2007). The concept of *cures* in epidemic routing serves to reduce total transmissions and overhead, by spreading delivery notification messages through the network (Harras, Almeroth, and Belding-Royer, 2005). A cure consists of a message hash and a timestamp. When a cure is received, the OLPC knows that the associated message has been delivered to the Ushahidi server. *Cured* messages will no longer be retransmitted to other XOs. By using a hash derived from the message contents as the key, we ensure that all copies of a message are cured. Lists of *cured* messages are exchanged any time messages are exchanged, in order to halt the flood of redundant messages through the network.

Exchanging Messages with Ushahidi

When an XO is connected to the Internet, it communicates with an Ushahidi server via an HTTP JSON API. Messages uploaded to the Ushahidi server will be geolocated and displayed on a web-based map, increasing situational awareness for first responders and any person with internet access. Ushahidi allows for developers to

extend its functionality via modular plugins. We created a plugin that creates a cure whenever a message is uploaded, and exchanges data between the OLPC and Ushahidi server as efficiently as possible. As mentioned earlier, we store each uploaded message hash with the time the message was received as a *cure* in a separate MySQL table. By extending Ushahidi's RESTful API with our plugin, we enable users to query this table via HTTP. When an OLPC obtains Internet connectivity, it obtains this list of cures from the server, then uploads all the messages on its flash storage that are not cured. The server can then send the OLPC any messages it does not already have. This method of data exchange minimizes or eliminates the exchange of duplicate messages between OLPCs and the Ushahidi server. This is particularly vital in areas where an entire community may share one satellite or other low-bandwidth link. Downloaded messages and cures are then distributed to nearby XOs using the mesh network. The injection of cures into the network by the Ushahidi server reduces unnecessary retransmission of already delivered messages in the mesh network.

TESTING

Our testing goals were to determine how well our activity shares messages and cures, the effective range at which OLPCs can communicate with each other, how effective our application is at bridging multiple groups of OLPCs, and the network overhead involved in epidemic message passing between nodes. Our testbed consisted of 8 XO-1.5 laptops joined in a mesh network. The XO laptop uses an 802.11b/g chipset and sets its default transmission rate at 1 Mbps in mesh mode. Thus, all tests were performed at 1 Mbps. Each device uses GNU/Linux (Fedora) for an operating system with kernel 2.6.31. We used several software tools during testing. To collect network data, we set up a non-XO laptop within range of all XOs and captured traffic with tcpdump. To inject messages into the network, we modified our activity to programmatically create 100 or 500 random messages of approximately 555 bytes each.

For the first test (Test 1), we injected 100 messages into the network from one machine and monitored network traffic until each machine received all messages. The second test (Test 2) was identical to the first, except that we increased the number of messages to 500. For our final test (Test 3), we tested range and mobility indoors and outdoors to determine the effective range of the OLPC mesh network.

Testing Results

Our experimental results show that our application is capable of sharing messages between OLPC users, but suffers performance problems due to protocol overhead from Telepathy Salut. Our tests show the OLPC's wireless radio range is excellent in areas with few obstructions, while still being acceptable in a campus setting where multi-path fading and large obstacles may reduce signal quality. Finally, our testing reveals the limitations of the OLPC's Sugar platform for message sharing and collaboration.

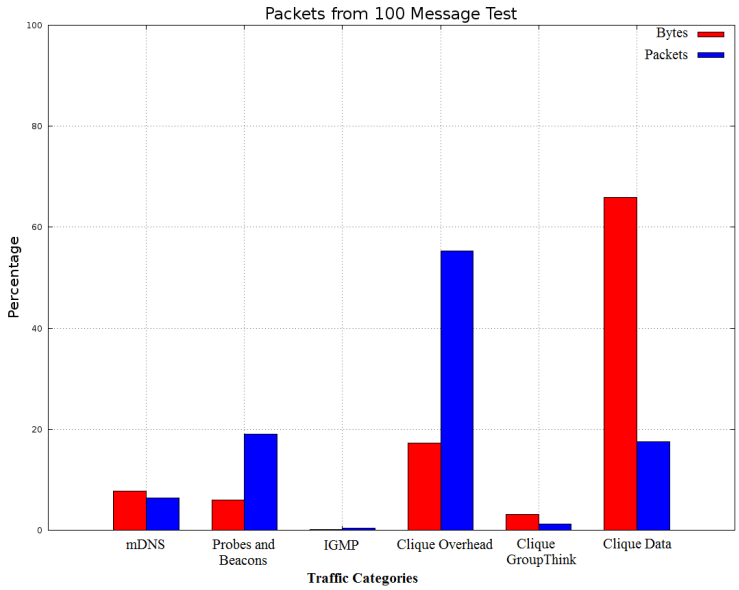

Figure 1. 100 Message Test Traffic

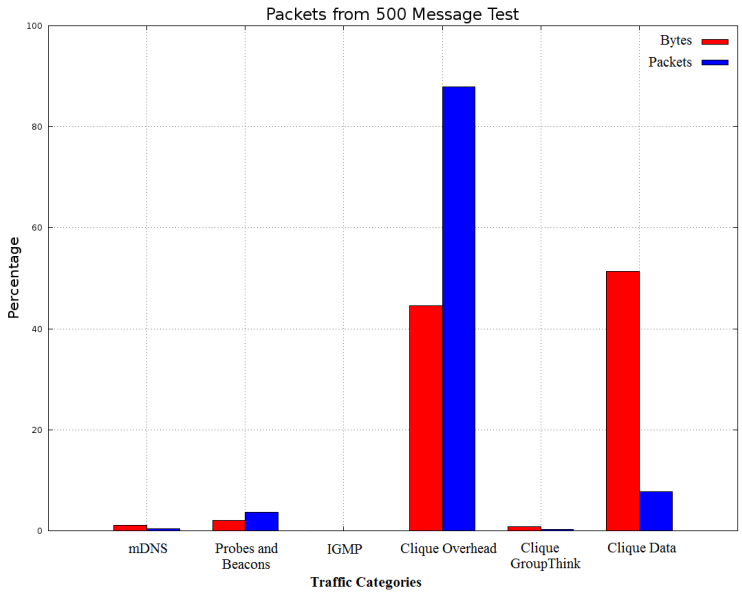

Figure 2. 500 Message Test Traffic

While Test 1 and Test 2 differ only in message batch size, the synchronization success ratio and network packet analysis differ significantly. Each message is approximately 555 bytes for either test, to represent a short test message and associated meta data. In Test 1, all 8 XOs received all 100 messages, whereas in Test 2 only 5 of 8 successfully received all messages. Our results show that when increasing message count from 100 to 500, Telepathy Salut overhead, specifically the Clique protocol, increases dramatically. We expected that overhead would increase due to the inefficiencies of epidemic routing, as contention, congestion, and collisions increased. However the observed increase in overhead can be attributed to the amount of traffic the Clique protocol generates to negotiate message passing.

Figure 1 shows the traffic breakdown from Test 1 as a percentage of total bytes and total number of packets (frames). We categorized the traffic into 6 buckets and measured bytes and number of packets over the duration of the test. The first three buckets are multicast-DNS, 802.11 network probes and responses, and Internet Group Management Protocol (IGMP), which is used for multicast. The next three buckets represent the three major components of our epidemic messaging implementation. Clique Overhead includes negotiation between clients to set up Telepathy *tubes*. As mentioned earlier, these tubes allow for inter-machine communication. The Clique GroupThink bucket shows traffic related to establishing and maintaining state synchronization for the shared Python dictionary we use to share messages. Lastly, Clique Data is largely composed of our messages.

Clique Data, packets containing parts of messages, account for the largest percentage of bytes, but less than 20% of overall packets. An unacceptably high percentage of both packet transmissions and throughput were used for Clique Overhead or negotiation packets. These overhead packets represent *over 50%* of the overall packets.

Figure 2 shows that in the 500 Message test, Clique Overhead is responsible for over 87% of the total packets, a striking increase. As observed in our 500 message test, the establishment and maintenance of inter-machine communication was extremely inefficient. In this test, of 788,884 total packets, 693,655 were Clique packets smaller than 120 bytes. Each packet contained 14 bytes or less of usable data, which represents at least *88.3%* overhead per packet. We suspect this severe overhead is the main cause of only 5 of 8 nodes successfully receiving all messages in Test 2.

Range and Mobility

Test 3 showed XOs are capable of communication with line of sight at a range of up to 350 meters. Our maximum successful transmission range in areas with many obstructions and potential causes of multi-path fading (such as concrete buildings, trees, people, brick walls, and a building emergency power generator) was 96 meters. The XO's twin antennas likely contributed to the excellent Wi-Fi range we observed.

This test also demonstrated the potential of mobile nodes to bridge disjoint mesh networks. Our single mobile node began the test as a member of one mesh network of OLPCs (group A). The mobile node received a copy of all messages from OLPCs in group A, then left the area covered by group A's mesh network (by going out of Wi-Fi range of every member in group A). When the XO moved within range of group B, it joined group B's

shared activity. All messages from group A were broadcast to every node in group B. Whenever any member of group B had Internet access, these messages were immediately uploaded to Ushahidi. The cures injected by the Ushahidi server were then shared with all nodes in group B (including the mobile node). When our mobile node returned to the area of group A, it brought with it a cure for each message, which included the exact time of receipt by the Ushahidi server. The cures successfully halt sharing of delivered messages to additional OLPCs.

DISCUSSION AND FUTURE WORK

Deploying our application on OLPC devices allows us to reach over 2.5 million students and teachers on 6 continents. Therefore a vital area of future work centers on improving usability for a worldwide population of children. Our current implementation only accepts typed messages, which is not ideal in areas of low literacy. In the future, our activity will be extended to include audio recordings and images instead of only text. These can be recorded using the OLPC's built-in microphone and camera. This would enable the use of this application even by individuals with limited literacy. Ushahidi already supports these formats, so our plugin could easily be extended to accommodate this.

Adding public key cryptographic signatures, while simultaneously linking a user's professional or social associations with their public key, could prove extremely useful to prioritize retransmission of important messages and evaluate the veracity of messages. Bundling the public keys of well-known humanitarian and government organizations would allow for intelligent message prioritization. For example, the Red Cross could use their private key to create a signed token, designating Bob as a Red Cross employee. Messages from Bob could then be given priority retransmission, or the particular collection of signed tokens associated with a user could give an indication of the trustworthiness of that user and the veracity of their information. For example, a user would be more likely to trust a message from Bob if they knew he was a Red Cross employee, or a user might place more trust in messages from a stranger Alice because the user and Alice have eight Facebook friends in common.

While communication between OLPCs can be very useful, adopting a cross-platform mesh networking solution would be beneficial. In particular, since OLPCs are simply Linux computers, utilizing the Serval or Commotion mesh networking projects would enable OLPCs to be part of a heterogeneous mesh of diverse devices, including OpenWRT routers and Android phones. A change in the mechanisms that enable mesh networking has the added performance benefit of removing the activity's dependence on Telepathy Salut and the Clique protocol, greatly reducing overhead.

ACKNOWLEDGEMENTS

We extend our thanks to Benjamin Schwartz, creator of the GroupThink library. We also thank Danielle Harlow and Hannah Anderson of the UCSB Education Department, and Danny Hembree and Caryl Bigngho of OLPC SoCal for helping us obtain XO laptops for development and testing purposes.

REFERENCES

1. Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., Scott, J. (2006). Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms. In Proceedings of INFOCOM 2006.
2. Harras, K., Almeroth, K., & Belding-Royer, E. (2005). Delay tolerant mobile networks (dtmns): Controlled flooding in sparse mobile networks. NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems, 41-91.
3. Hughes, A., Palen, L., Sutton, J., Liu, S., and Vieweg, S. (2008) "Site-Seeing" in Disaster: An Examination of On-Line Social Convergence. In Proceedings of ISCRAM 2008.
4. Palen, L., Hiltz, S., and Liu, S. (2007). Online Forums Supporting Grassroots Participation in Emergency Preparedness and Response. Communications of the ACM, 50 (3) (Mar. 2007): 54-58
5. Song, L., & Kotz, D. F. (2007). Evaluating opportunistic routing protocols with large realistic contact traces. In Proceedings of the second ACM workshop on Challenged networks (pp. 35-42). ACM.
6. Vahdat, A., & Becker, D. (2000). Epidemic routing for partially connected ad hoc networks (p. 18).