Int. J. Communications, Network and System Sciences, **2012, 5, 496-504**
http://dx.doi.org/10.4236/ijcns.2012.58060 Published Online August 2012 (http://www.SciRP.org/journal/ijcns)

MeshMS: Ad Hoc Data Transfer within Mesh Network

Paul Gardner-Stephen[1], Jeremy Lakeman[1], Romana Challans[1], Corey Wallis[1], Ariel Stulman[2], Yoram Haddad[2,3]
[1]Flinders University of South Australia, Bedford Park, Adelaide, Australia
[2]Jerusalem College of Technology, Jerusalem, Israel
[3]Ben Gurion University of the Negev, Beer-Sheva, Israel
Email: paul@servalproject.org, jeremy@servalproject.org, romana@servalproject.org, corey@servalproject.org, stulman@jct.ac.il, haddad@jct.ac.il

Received July 4, 2012; revised August 2, 2012; accepted August 20, 2012

ABSTRACT

File and data distribution can be easily classified as one of the basic uses of networks. With uses ranging from Short Message Service (SMS) to program updates, from micro-blogging to social networking, every network today must support some type of file and data dissemination method. Infrastructure networks have already implemented these services using well known communication protocols. Ad hoc networks pose a greater challenge due to their sporadic network set-up. At a given time we do not know who is connected to the network, and whether the intended recipient of the data can be reached. In this paper we introduce Serval MeshMS, a protocol for ad hoc file and data distribution, enabling the diffusing of data through an ad hoc mesh network. It is based on a single-hop, store and disseminate opportunistic architecture, and has been shown to work over great distances. Preliminary implementations are encouraging, with surprising results achieved.

Keywords: Instant Messaging; Mobile Mesh Networks

1. Introduction

There are many types of networks in existence today. A common task that they must all be capable of performing is the ability of transferring data. A few bits of data (such as SMS) actually take up a large portion of bandwidth of today cellular networks. According to ITU figures [1], the total number of SMS sent globally tripled between 2007 and 2010, from an estimated 1.8 trillion to a staggering 6.1 trillion. That is almost to 200,000 text messages sent every second. Many services and applications have developed around this technology, with recent utilization in the medical field as well [2]. Ad hoc mesh networks must also cope with these types of data transfers, while overcoming challenges pertaining specifically to these types of networks.

A MeshMS file and data distribution protocol must provide a platform that enables blocks of data and associated meta-data to be diffused through an ad hoc mesh network. These protocols use a "one-hop" communication scheme as they don't necessarily see their final destination, and must take advantage of a "store-and-forward" mechanism to overcome the time/space problem encountered in ad hoc mesh networks. That is, the protocol must provide primitives for transferring to active neighbors an original or a received bundle, including data and meta-data, storing bundles for future dissemination, and discerning when a new neighbor that hasn't received the bundle entered into direct communication range. When performed by all nodes in the mesh, a bundle will, over time, replicate to all nodes in the mesh regardless of their distance from the sending node. Of course, node-specific or network-wide policies must be instituted for repetitious, outdated (aging) or over-sized bundles, in order to ensure maximum available bandwidth and avoid degradation of the overall network quality of service (QoS).

In this paper we present an experience report regarding the development of a MeshMS for the Serval [3] ad hoc mesh network. We use a flooding algorithm to forward messages to all nodes in the network, and offer the capability for a store-and-forward mechanism to create a delay-tolerant communication medium.

The remainder of this paper is organized as follows: Section 2 describes how a mesh-based short-message-service might be constructed. Section 3 describes some surprising results of the current implementation. In Section 4 we propose future research directions, and present some potential benefits such a MeshMS service might offer for mobile ad hoc networks. We also describe how it can be generalized to offer related services, such as crowd-sourced information and Twitter™ style micro-

blogging. A conclusion is presented in Section 5.

Related Work

The great potential of ad hoc networks has been identified by the research community long ago. Already in 1994, some seminal papers presented issues specific to ad hoc networks; [4] listed some of the challenges in what was then a "new" area, and [5] presented the famous paper on the routing issue. Most of the literature from that point on focused on the routing issue, with dozens of protocols presented from 1995 in [6] until [7,8], and references therein.

Meanwhile IETF [9] and its MANET (Mobile ad hoc Network) working group devoted prolific efforts on the routing issues [10-12] together with some general protocol aspects such as addressing [13] and packet format [14]. Efficient flooding protocols which allow for better utilization of the limited radio spectrum while taking the need for energy saving for the mobile devices into account has also been considered in [15-18] and references therein. However, so far most of the work proposed has been only theoretic; at most validated through some simulation.

The problem of the inaccuracy of validation especially in wireless network has been raised by [19]. In [20], "incredible" simulation results in mobile ad hoc network have been listed. The need for an international standard "controlled" simulator is still an open issue in the field of wireless networks [21].

Some experimental testbeds [22] have been deployed, but this is mainly for wireless mesh networks that assume an access point is available among the mobile end-point devices which is considered as infrastructure mode. For the specific case of ad hoc mode, much fewer experimental testbeds have been built [23,24]. In addition some projects were started to provide ad hoc wireless mesh networks to the public [25-27], but none of them are as accessible as Serval which is an application for smart phone devices.

In this paper we present a real testbed deployed with new MeshMS service, which is SMS service adapted for ad hoc networks. For the purpose of our experiments, we describe a protocol which allows efficient transmission of the information through incremental download of only required information at intermediate nodes, minimizing repeated useless traffic. The system described in this paper retains its functionality even in highly-partitioned networks having only occasional connectivity. It has been implemented on Android enabled smart-phones and accepted for distribution on the Android Market. This allows for mass distribution, providing the much needed critical mass for such an architecture to succeed "in the wild".

2. Constructing an SMS

Every SMS technology must consist of a mechanism allowing for one user to post a message to another user. In MeshMS this is accomplished by having each user's device maintain a public log of all messages that they have sent; thus, posting a message is reduced to adding the message to this public log. This log is then replicated and distributed as a data bundle by other nodes on the network. The receiver will, hopefully, at some later time, receive a replicate of the sender's public message log, and discover that a message has been sent to it. The message is then extracted from the log and displayed to the intended recipient user.

Confirmation of delivery is achieved by sending a reply message via the same mechanism. This provides the sender with confirmation that the message was successfully received and that it need not be sent again. Clearly, confirmation of delivery messages must not be confirmed so as to avoid an infinite loop and bloating of network traffic.

2.1. Format of the Public Message Log

1) *Message Length Field*: As heads of message logs will be discarded as soon as possible for efficiency reasons, logs must have the ability of traversing from the tail backwards. To facilitate this ability, a length field for each message is appended to it's end. **Table 1** describes the format of this length field, a format that was designed with the intent of keeping small messages small while still allowing for large messages as well. To simplify the length encoding process, and in some instances save an additional byte, the length field does not include it's own length.

2) *Message Body*: The message body consists of a one byte version code, being 0x01 for messages sent in clear text, followed by one or more fields, each of which consists of an identifier and the field body. If the identifier has bits 6 and 7 clear, then the field body is a nullterminated string, unless the identifier is <0x10, in which case it is a 32 byte binary string. If the identifier has bit 6 set and bit 7 clear, then the field body is a binary extent whose length is indicated by two bytes immediately following the identifier. If the identifier has bit 7 set, and bit 6 clear, then the field body is a binary extent whose length is indicated by four bytes immediately following the identifier. The initial set of valid field identifiers is listed in **Table 2**.

To ensure efficient scanning of messages by potential recipients, it is required that recipient addresses, whether expressed as telephone numbers, subscriber IDs or otherwise, are the first fields of the message. Messages not adhering to this format may be, at the discretion of node implementation, ignored or assumed to be broadcast

Table 1. Message length field format.

Length	n − 5	n − 4	n − 3	n − 2	n − 1	n
1 - 255	-	-	-	-	-	0 - 254
256 - 505	-	-	-	-	0-0xf9	0xff
506 - 761	-	-	-	0 - 255	0xfa	0xff
762 - 65,535	-	-	bits 8 - 15	bits 0 - 7	0xfb	0xff
>65,535	bits 24 - 31	bits 16 - 23	bits 8 - 15	bits 0 - 7	0xfc	0xff

Table 2. Message field identifiers.

Identifier	Field Description
0x00	Recipient SID
0x10	Recipient telephone number
0x11	Sender telephone number
0x3e	Message body (null terminated string, unlimited length, e.g., SMS, email)
0x7e	Message body (binary string up to 64 KB, e.g., Unicode SMS or MMS, short voice message)
0x7f	Message Signature
0xbe	Message body (binary string up to 4 GB, e.g., file, video message, long voice message)

messages available for reception by all parties, a feature that is leveraged to facilitate micro-blogging (see Section 4.3). Note that multiple recipients can be specified for a single message by including multiple recipient fields.

3) *Message Cipher Block*: If the message body version code byte is >0x7f, then the block is encrypted. If the code is >0xbf, then the block and the destination address are encrypted. If the code is between 0x80 and 0xbf, then the destination address (as expressed using the appropriate message body field) is in the clear, but the message body is encrypted using the specified cipher. The codes for CryptoBox Curve25519 authenticated encryption [28] are 0x80 and 0xc0 for ciphered body and ciphered body and destination, respectively. The format of encrypted fields and messages is dependent on the particular encryption schemes. For encrypted message bodies, the encrypted data format is contained entirely within the message body field. For messages with encrypted recipient, the entire message following the message body version code is treated as a single encrypted unit, which is then decrypted by the authorized recipient and treated as a plain text message block.

Messages sent with destination address encryption do not guarantee delivery, as nodes have the option of declining such messages. This is incorporated into the protocol, as such blind messages must be decrypted to see if they are addressed to the decrypting node, an operation carrying a substantial computational cost. Destination address encryption is intended solely for communication between parties who have agreed to receive such communications from one another, and are willing to accept the increased energy cost that this potentially more private mode of communications entails.

4) *Example Plain Text Message Block*: To illustrate how these components of a message are assembled, consider the clear text message "Buy casaba melons on the way home" sent to telephone number +18005552600. **Figure 1** shows a possible format of this message, constructed by concatenating the message version code (01), the recipient telephone number field encoded in null-terminated ASCII (10 2b 31 38 30 30 35 35 35 32 36 30 30 00), and the message body itself, also as a null-terminated string (3e 42 75 79 20 63 61 73 61 62 61 20 6d 65 6c 6f 6e 73 20 6f 6e 20 74 68 65 20 77 61 79 20 68 6f 6d 65 00). In this instance the message is not being signed, so all that remains is to append the length field, which is 50 bytes. Since the message length is less than 256 bytes, a single byte will suffice to encode the length (0x31). The length code is 0x31 and not 0x32, because the length field encoding scheme of **Table 1** takes advantage of the fact that messages must contain at least a version code and therefore cannot be of zero length.

2.2. Other Considerations

1) *Minimization of Network Traffic and Data Storage*: Each time a user sends a new message, their public message log file will grow. Initially, this file will be small, so it does not introduce great inefficiency into the system. If,


```
01 10 2b 31 38 30 30 35  .+18005
35 35 32 36 30 30 00 3e  552600.>
42 75 79 20 63 61 73 61  Buy casa
62 61 20 6d 65 6c 6f 6e  ba melon
73 20 6f 6e 20 74 68 65  s on the
20 77 61 79 20 68 6f 6d   way hom
65 00 31                 e.1
```

Figure 1. Example plain-text message block showing both hexadecimal and ASCII representation of version code (green), recipient telephone number (yellow), message body (teal) and length (light blue) fields. Non-printable ASCII characters are depicted by a period.

however, a mechanism for removal of log entries does not exist, over time, the accumulation of sent messages will cause the log file to grow. Stale messages long received by their intended recipient will be continuously sent, introducing heavy storage and transmission burden on nodes. In addition, the intended recipient must keep track of which messages are already stale (although received again in a new transmission) so as to not present them to the user redundantly.

The approach we took attempts to transmit which part of the log file floating among the nodes in the network is new. This is accomplished by indicating in the log file manifest the position of the first byte in the message log that cannot yet be discarded; thus, allowing each node in the network to discard stale content. This greatly reduces the size of the data bundles that are replicated in the network, mitigating a possible needless traffic bloat. In addition, stale messages, ones that have exceeded a timeout threshold, even though they were not confirmed and messages with delivery confirmation are also removed from the log file.

To implement this, the following enhancements to the protocol are required:

1) The log file manifest requires a field, *stream_offset* that indicates the first byte position in the data stream that is still part of this version of the file.

2) The hash of the file stored in the manifest to allow verification of the associated file. This hash should include only the data from *stream_offset* onwards. This is necessary so as to allow for nodes to discard stale data and still verify the received log file.

3) When a node begins the retrieval of a log file, it must first examine whether there is an overlap between the log file pending retrieval and an older version it already has. If no earlier version exists, the log file is retrieved. If a previous version exists and there is an overlap, only the non-overlapped data is downloaded and appended to the existing log file on the node. Any deprecated portion of the stream should be discarded.

4) A distributing node offers the ability to specify which part of the log file is requested, and provide the means for downloading that specific portion.

To illustrate how these measures work to minimize storage and network transmissions, an example for message exchanges between two nodes, Alice and Bob, is shown in **Figure 2**. We assume that each node periodically polls the neighboring nodes to discover any updated manifest files. In addition, we assume that Alice's public message log version is at 0 (nothing has ever been sent), and Bob's log is at 4.

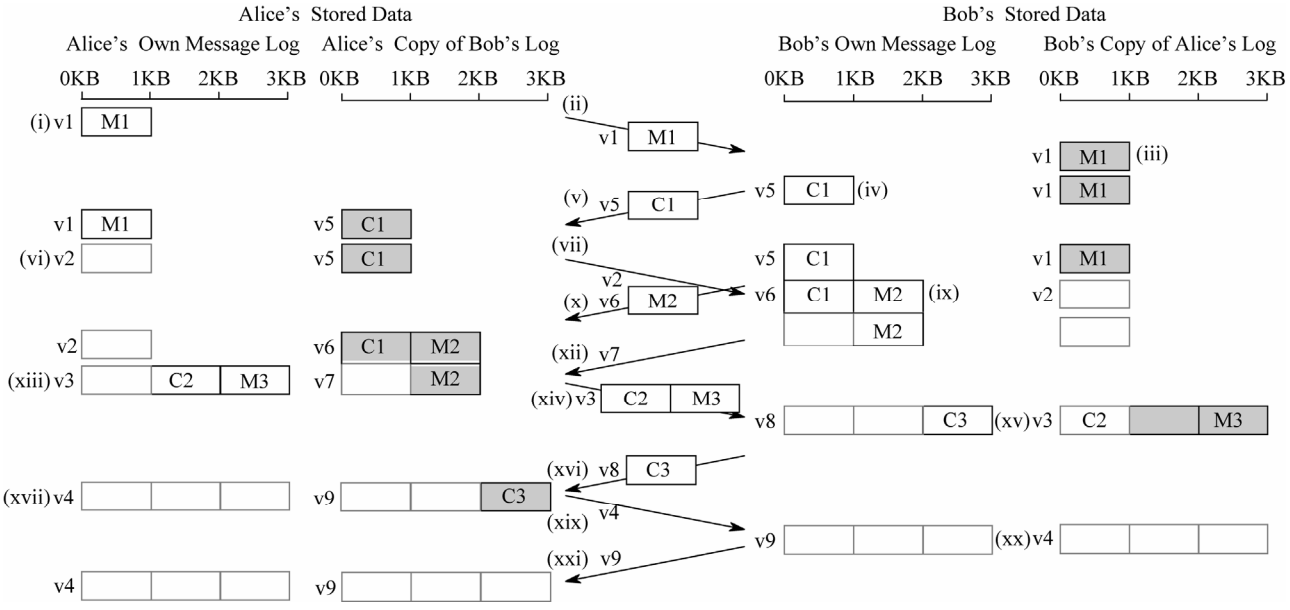

Figure 2. Example of using *stream_offset* to minimize network traffic & storage requirements. Greyed boxes indicate replicas of message logs from other nodes. Empty boxes represent data that has been discarded by the respective party on discovering that it has become stale.

Alice begins by posting a message, M1, to Bob. This is accomplished by appending it to Alice's own public message log, and setting a version variable for her message log to 1 (1). At some point Bob discovers through one of his neighbors that Alice's message log is now at version 1 which differs then what he has stored for her, and fetches the updated version (2). He can then read the message. A copy of the message log is stored locally, and updates his record for the last known version of Alice's message log (3). Bob also generates a confirmation, C1, by appending that confirmation to the end of his own message log. This also updates his log file version to 5 (4). As the confirmation is new, its transmission and storage is required. When Alice hears of Bob's new message log she fetches it from some neighbor, resulting in the transmission of the confirmation C1 (5).

At this point Alice knows that Bob has received message M1, so any future version of her message log is no longer required to include this message. By advancing the *stream_offset* field in her log manifest to the first byte after M1 allows her to mark M1 as stale. She must then increase the version field in the manifest file to 2 (6) to announce the change in her log file.

Before Bob notices that Alice's message log version has changed, he sends a reply, M2, to Alice's message M1 by appending it to his message log and updating the log version to 6 (9). Alice, upon noticing the increased version number of Bob's message log, fetches M2 (10), but without the need to repeat the transfer of confirmation C1, because Alice knows she already has that part of Bob's message log.

Meanwhile, Bob has finally noticed that Alice's message log version is now version 2 (7). As version 2 indicates that the message log is no longer than it was in version 1, Bob does not need to transfer any data. Instead, by learning that the existing content of Alice's message log is stale, he can free the space that it previously occupied. Further, Bob now knows that his confirmation C1 has been received by Alice, and so Bob marks that part of his message log as stale and increases his log version to 7 (11).

Alice notices that Bob's version has increased to 7 (12), but as Bob's message log has not grown, does not need to fetch any data. Rather, the discovery that Bob marked part of his message log as stale, allows her to free the matching portion of her copy of Bob's message log (13). At the same time that Alice generates a confirmation, C2, to Bob's message, M2, she also creates a new message, M3, to Bob. Both are appended both to her message log, and the version is increased to 3 (13).

In time Bob discovers that Alice's message log version has increased to 3, and downloads the new content (14). Again, Bob does not need to transfer nor retain storage for the stale message M1. Bob generates a confirmation C3, for message M3, and appends it to his message log. Bob also knows by receiving confirmation C2 that message M2 has been received, and can thus mark it as stale in his public message log. Both changes are reflected in version 8 of his message log (15).

Alice soon discovers that Bob's message log version has increased, and fetches the new data (16). Upon hearing confirmation C3, Alice is able to mark message M3 as stale in her message log, which increases the version to 4 as a result (17). Bob can then mark confirmation C3 as stale, and update his message log version to 9, to reflect this change (20).

Finally, Alice discovers the new version of Bob's message log (21), and without having to transfer any part of the message log itself, she is able to free the storage associated with C3. Thus at the end of the message exchange, both parties are able to completely free all storage associated with the messages and confirmation.

As this simple example shows, the ability to signal the rapid deprecation of messages known to have been received, provides the basis for avoidance of mesh-wide distribution of stale messages. This is particularly important when the number of nodes in the mesh network increase, diminishing useless network traffic; mitigating one barrier of scalability. It is acknowledged, however, that as the number of parties engaged in intertwined conversations grows the more difficult it is to cancel stale message. The *stream_offset* value cannot advance based on the reply of a single message recipient, as an earlier unacknowledged message may exist that still needs to be disseminated. Explicit cancel messages might have the ability to aid in this area, but such explorations are left as a future research.

2) *Security*: Section 2.1 above provides explanation as to how a short message service may be constructed on the bundle dissemination primitives of the protocol. To be a useful service, however, it is necessary to provide an appropriate security environment. The main security issues currently under research and development are authenticity, integrity and confidentiality. Since they have not been incorporated as of date into Serval, we defer their discussion to future papers.

2.3. Implementation

The data distribution service was built as a simple HTTP based service where all log files and their associated manifests are presented in a single HTTP directory listing. Nodes then periodically query this service on their peers, download all manifest files and then decide which log files to download. In the current implementation the decision consists solely of downloading all log files that are either new to the receiver or a newer version of an existing file.

Serval first attempts to deliver a message directly to the recipient device if it is reachable on the mesh at the time of sending. If the recipient is not reachable on the mesh at the time of sending, Serval resorts to the "store-and-forward" mechanism described in this paper. This policy was chosen to minimize the size of public message logs, and to ensure rapid reception of any given message.

The MeshMS was integrated into the Serval [3] software for Android [29] based phones. As a convenience, the existing WebSMS application for Android was used for Android integration, as it already offers a user-interface and well-defined API for sending SMS-like messages. The necessary routines are called to create an appropriate message block, append it to the end of the subscriber's public message log, and notify the software that the file had changed. Reception of messages was implemented by placing hooks into the service software that called the necessary routines to examine a public message log and check for messages addressed to the inspecting node. The last offset in any given public message log was remembered to prevent duplicate parsing of portions of a message log already seen.

As the intent of this initial implementation was to demonstrate the feasibility of the technology, security primitives such as encrypted and digitally signed message logs were deferred to a later version. Also, neither the algorithms for expiring old portions of public message logs nor for acknowledging the reception of messages were implemented.

3. Results & Discussion

The file distribution software was first tested by importing a number of files, ranging from a few bytes to megabytes, on a dozen Huawei IDEOS U8180 handsets. Due to the high WiFi data rates of up to 72 mega-bits per second possible when the phones came into close proximity, files were replicated rapidly. The ability of the service to successfully store-and-forward messages over extended times and distances was accidentally established when one of the phones was not emptied of data bundles when it was carried from Australia to South Africa for the testing of the MeshMS service, delivering a 2 MB photograph of Serval's Adelaide laboratory to the team in Magaliesburg, approximately two hours drive from Johannesburg, South Africa. Thus, the message traveled a total distance of ~10,000 km with a delivery time of approximately five days (~30 seconds attributed to the MeshMS protocols, and the remainder being the physical transit of the courier phone between Africa and Australia).

The MeshMS service was also tested with a team of 14 individuals who were technology literate, but had not received prior training in the use of the software nor were they directly involved in it's development. Within minutes they were successfully sending short messages amongst themselves, with messages being delivered instantly. Use of the store-and-forward MeshMS service described in this paper was successfully achieved by disabling the Serval mesh software on the recipient's phone, and then reactivating it once the message had been dispatched.

The service was further tested by separating the sender and recipient of a message sufficiently so that they could not communicate directly. A third phone was carried by a person from within the communications range of the message sender towards the communication range of its intended recipient. The aim was that the courier node should replicate the public message log of the sender, and make it available for replication, and hence reception, by the final recipient. This was indeed achieved, with the message being received by the recipient before the person carrying the courier node approached the final recipient; thus, indicating the delivery had indeed occurred soon after the courier phone and final recipient's phone came into mesh communications range. No special actions were required on the part of the courier, recipient or sender to facilitate each of the transfers involved. The sender sent the message as normal, and courier simply walked between the nodes to create an occasional connection, and the recipient was notified of the reception of a new message.

Based on the success of these two tests, a message was then sent from a phone located in South Africa to another phone located in Australia, with the first author carrying the courier phone on his journey to Australia. The intent was of facilitating the infrastructure-free delivery of a short message over inter-continental distances. The message passed through three phones to achieve it's delivery, with each transfer occurring automatically and without user intervention.[1]

The demonstration of this capability is significant, as it establishes the potential of an appropriately designed mesh network to not only deliver mesh-local traffic, but to carry traffic between more distant parties, parties who may never have a direct real-time connection with each other. Thus the MeshMS represents a means to facilitate mesh-mediated communications, even when a mesh suffers from acute and chronic partitioning.

There are many use-cases where this capability is potentially beneficial. For example, in developing countries it is not uncommon for a number of people from a given village to go to the same city in search of work. In such cases, sporadic traffic between the city and the village is likely. If the people traveling between the two locations

[1]A video documenting the process has been made available at http://www. youtube.com/watch?v=KDhJcwsnxf0

carry phones running Serval MeshMS software, they will, without any special action on their part, facilitate the exchange of messages between the two locations. More generally, MeshMS can be used to enable communications between any disparate communities as long as there there is some traffic, even sporadic, between them. In addition, populations within a given locale who simply wish to avoid the excessive charges levied by carriers for the delivery of short messages, can use this service with great assurance and reasonably high QoS. Moreover, message exchange via MeshMS is not limited to short messages, but can also include voice messages, photographs, video, and other forms of information.

4. Future Directions

4.1. Protocol Improvements

At present the protocol requires nodes to actively poll other nodes on the mesh. This results in both increased latency and data transfer, due to the periodic polling. A possible better approach is to have the a Serval daemon advertise a selection of recently updated public message logs, allowing nodes to discover new content faster, and without active polling on their part, hopefully saving bandwidth.

The format of the web page that is used to fetch the list of message logs could also be improved to reduce its size, by having a binary version of the page that lists—in compact form—the version, size and offset information of each manifest. Implementing the above suggestions, in conjunction with transitioning the public message log distribution from HTTP, which is inherently unicast, to a broadcast-aware mesh protocol that allows multiple nodes to receive a single public message log transmission, will save bandwidth. Since the entire system is working above WiFi, some care must be taken to maximize performance. Unicast WiFi packets are acknowledged, and are automatically retransmitted by the WiFi chipset, and can be transmitted at any WiFi bit rate. In contrast, broadcast WiFi packets are not acknowledged, retried, or able to be transmitted at any speed other than 1 mbit/second [30]. Thus better implementation QoS can possibly be implemented by using a unicast message to a single node, which can occur at full WiFi speed, but with other nodes listening in to network traffic and actively logging what they hear. Specifics are left to future research.

1) Localizing Traffic: The current specification of the public message log format will result in the system attempting to replicate every public message log globally. This clearly does not scale. Thus it is intended to implement limitations on the distribution reach of the public message logs by various means. One approach is to have a time-to-live (TTL) descriptor similar to the Internet Protocol [31], so that messages tend to distribute with a limited range. Further possible refinements include filtering based on geographic information or prioritization of senders who are in a node's social network, such that locally relevant traffic takes priority over stray traffic from parties unknown to a given local network.

2) Improving the Public Message Log Format: While the present work has demonstrated the feasibility of the MeshMS system, extensive improvements to the underlying file structures and semantics are possible. For instance, the addition of application associations at the MeshMS message level, will allow messages to be intelligently routed to applications. This would support the interactive education or disaster response coordination use cases by allowing messages produced by such applications to be addressed to the corresponding application on another node, freeing such applications from having to poll the MeshMS system, or filter through all arriving messages themselves.

It is also intended to optionally tag each part of a MeshMS with mime-type and/or file-name attributes to allow for easier decoding of the message on the recipients end. Combining this with application association, will allow for a variety of flexible and interactive store-and-forward-based services. For example, web browsing through a proxy, where the entire web page to be displayed is encapsulated in a MeshMS and directed to a web browser automatically upon the receipt or the opening of the MeshMS message.

3) Implementing and testing Cryptographic Protocols: As previously mentioned (see 2.2), security protocols must be defined, implemented and tested by the Serval security team. Only after rigorous testing, including examination against known vulnerabilities, can they be incorporation into the Serval ecosystem.

4.2. Supporting Education

A variation on the use cases previously described is the use of this technology to distribute educational material, such as text books, interactive educational materials and student results and feedback. This approach could allow, for example, a teacher to set an in-class exercise and monitor, in near-real-time, the progress of each student through the work. This would allow teachers to, amongst other things, identify students who are not tackling the material or are struggling to make satisfactory process as these issues occur, rather than at the end of the session when it would be too late for immediate remedial actions. A key innovation in this potential application is that there is no supporting infrastructure or administration required in the school, and no data distribution costs, making it more feasibly in developing and remote locations than more traditional infrastructure-oriented approaches to

computer mediated learning.

4.3. Generalizing MeshMS to Support Infrastructure-Free Micro-Blogging

As the MeshMS service distributes messages in a public message log, it is trivial to adapt the system to create a Twitter™-like micro-blogging service, simply by sending broadcast messages, and some marker indicating that the messages are intended to be treated as micro-blogs. Indeed, such a system, Serval Meow, is planned and scheduled for implementation and integration into the Serval software system. This system has been designed to make it relatively easy to gateway between the Twitter™ and Meow universes, allowing the bidirectional exchange of Twitter™ tweets and Serval mews, and thus with other Twitter™-connected services such as the Ushahidi [32] situational-awareness tool. This has significance for the use of Meow in disaster relief settings where Twitter™ is unavailable in the disaster theater due to loss of infrastructure, but where micro-blogs have considerable value in helping both internal and external organizations coordinate and optimize their responses.

4.4. Trial Exercise with a Humanitarian Disaster Response Organization

As a result of the successful demonstration of the store-and-forward MeshMS service described in this paper, the Serval Project has been invited to participate in a trial exercise in disaster response in 2012, by providing various MeshMS-based services for use by up to 50 participants. It is expected that this exercise will provide valuable feedback and data on the use and behavior of the system in a realistic scenario, which will be fed back into the ongoing development process. It will also be used by the relief organization to assess the MeshMS technology for possible future incorporation into their standard practices.

5. Conclusion

The Serval MeshMS system has successfully demonstrated the feasibility of delivering a message more than 10,000 km using only a store-and-forward protocol on Android based mobile telephones. This was accomplished without reliance on any supporting infrastructure, while still using a familiar SMS-like interface. The application of this approach has enormous benefits, and can be expanded in many possible ways. Use of delay-tolerant mesh communications can positively impact the delivery of communications and communications-oriented services and technologies in potentially difficult situations. These achievements are recognized by disaster response teams, and are the basis for the invitation to participate in a disaster-response training exercise.

6. Acknowledgements

The work in this paper was funded in part by the Shuttleworth Foundation, the Awesome Foundation (Boston), the NLnet foundation, and Serval Project Inc.

REFERENCES

[1] "The World in 2010—The Rise of 3G," 2010. http://www.itu.int/ITU-D/ict/material/FactsFigures2010.pdf

[2] R. T. Lester, P. Ritvo, E. J. Mills, A. Kariri, S. Karanja, M. H Chung, W. Jack, J. Habyarimana, M. Sadatsafavi, M. Najafzadeh, C. A. Marra, B. Estambale, E. Ngugi, T. B. Ball, L. Thabane, L. J. Gelmon, J. Kimani, M. Ackers and F. A. Plummer, "Effects of a Mobile Phone Short Message Service on Antiretroviral Treatment Adherence in Kenya (Weltel Kenya1): A Randomised Trial," *The Lancet*, Vol. 376, No. 9755, 2010, pp. 1838-1845. doi:10.1016/S0140-6736(10)61997-6

[3] The Serval Project, 2010. http://www.servalproject.org/

[4] D. B. Johnsorn, "Routing in Ad Hoc Networks of Mobile Hosts," 1*st Workshop on Mobile Computing Systems and Applications*, Santa Cruz, 8-9 December 1994, pp. 158-163. doi:10.1109/WMCSA.1994.33

[5] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, New York, 31 August-2 September 1994, pp. 234-244.

[6] P. Krishna, M. Chatterjee, N. H. Vaidya and D. K. Pradhan, "A Cluster-Based Approach for Routing in Ad-Hoc Networks," *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*, Berkeley, 1995, pp. 1-10.

[7] J. H. Luo, D. X. Ye, X. Liu and M. Y. Fan, "A Survey of Multicast Routing Protocols for Mobile Ad-Hoc Networks," *Communications Surveys Tutorials*, *IEEE*, Vol. 11, No. 1, 2009, pp. 78-91. doi:10.1109/SURV.2009.090107

[8] L. Chen and W. B. Heinzelman, "A Survey of Routing Protocols That Support Qos in Mobile Ad Hoc Networks," *Network*, *IEEE*, Vol. 21, No. 6, 2007, pp. 30-38. doi:10.1109/MNET.2007.4395108

[9] V. G. Cerf, "Report of the Second Ad Hoc Network Management Review Group," RFC 1109, 1989.

[10] A. Roy and M. Chandra, "Extensions to OSPF to Support Mobile Ad Hoc Networking," RFC 5820 (Experimental), 2010.

[11] C. Perkins, E. Belding-Royer and S. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), 2003.

[12] S. Corson and J. Macker, "Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," RFC 2501 (Informational),

1999.

[13] I. Chakeres, "IANA Allocations for Mobile Ad Hoc Network (MANET) Protocols," RFC 5498 (Proposed Standard), 2009.

[14] T. Clausen, C. Dearlove, J. Dean and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format," RFC 5444 (Proposed Standard), 2009.

[15] Y. J. Yi, M. Gerla and T. J. Kwon, "Efficient Flooding in Ad Hoc Networks: A Comparative Performance Study," *IEEE International Conference on Communications*, Vol. 2, 2003, pp. 1059-1063.

[16] W. Zhao, M. Ammar and E. Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, New York, 24-26 May 2004, pp. 187-198. doi:10.1145/989459.989483

[17] S. Pleisch, M. Balakrishnan, K. Birman and R. van Renesse, "Mistral: Efficient Flooding in Mobile Ad-Hoc Networks," *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, New York, 22-25 May 2006, pp. 1-12. doi:10.1145/1132905.1132907

[18] W. Chen, R. K. Guha, T. J. Kwon, J. Lee and Y.-Y. Hsu, "A Survey and Challenges in Routing and Data Dissemination in Vehicular Ad Hoc Networks," *Wireless Communications and Mobile Computing*, Vol. 11, No. 7, 2011, pp. 787-795. doi:10.1002/wcm.862

[19] T. R. Andel and A. Yasinsac, "On the Credibility of Manet Simulations," *Computer*, Vol. 39, No. 7, 2006, pp. 48-54. doi:10.1109/MC.2006.242

[20] S. Kurkowski, T. Camp and M. Colagrosso, "Manet Simulation Studies: The Incredibles," *SIGMOBILE Mobile Computing and Communications Review*, Vol. 9, 2005, pp. 50-61. doi:10.1145/1096166.1096174

[21] C. Partridge, "Forty Data Communications Research Questions," *SIGCOMM Computer Communication Review*, Vol. 41, No. 5, 2011, pp. 24-35. doi:10.1145/2043165.2043170

[22] I. F. Akyildiz, X. D. Wang, and W. L. Wang, "Wireless Mesh Networks: A Survey," *Computer Networks*, Vol. 47, 4, 2005, pp. 445-487. doi:10.1016/j.comnet.2004.12.001

[23] C. Tschudin, P. Gunningberg, H. Lundgren and E. Nordström, "Lessons from Experimental Manet Research," *Ad Hoc Networks*, Vol. 3, No. 2, 2005, pp. 221-233. doi:10.1016/j.adhoc.2004.07.007

[24] W. Kiess and M. Mauve, "A Survey on Real-World Implementations of Mobile Ad-Hoc Networks," *Ad Hoc Networks*, Vol. 5, No. 3, 2007, pp. 324-339. doi:10.1016/j.adhoc.2005.12.003

[25] Village Telco, 2011. http://villagetelco.org/

[26] Openbts. http://wush.net/trac/rangepublic

[27] Freedombox. http://www.freedomboxfoundation.org/

[28] D. J. Bernstein, T. Lange and P. Schwabe, "Nacl: Networking and Cryptography Library," 2012. http://nacl.cace-project.eu/

[29] Android. http://www.android.com/

[30] IEEE Std 802 Part 11, "Information Technology—Telecomunications and Information Exchange between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (Mac) and Physical Layer (Phy) Specifications," 1999.

[31] J. Postel, "Internet Protocol," RFC 791 (Standard), 1981.

[32] O. Okolloh, "Ushahidi, or 'Testimony': Web 2.0 Tools for Crowdsourcing Crisis Information," Participatory Learning and Action, 2009, pp. 65-70.

