Large-scale hybrid ad hoc network for mobile platforms: Challenges and Experiences

Nirmit Desai, Wendy Chong, Heather Achilles, Shahrokh Daijavad
IBM T. J. Watson Research Center
Yorktown Heights, NY, USA
{nirmit.desai, wendych, hachilles, shahrokh}@us.ibm.com

Thomas La Porta
Dept. of Computer Science and Engineering
Penn State University
University Park, PA, USA
tlp@cse.psu.edu

***Abstract*—Peer-to-peer (p2p) networks and Mobile ad hoc networks (MANET) have been widely studied. However, a real-world deployment for the masses has remained elusive. Ever-increasing density of mobile devices, especially in urban areas, has given rise to new applications of p2p communication. However, the modern smartphone platforms have limited support for such communications. Further, the issues of battery life, range, and trust remain unaddressed. A key question then is, what kinds of applications can the modern mobile platforms support and what challenges remain? This paper identifies a class of applications and presents a novel center-to-peer-to-peer (c2p2p) architecture called Mesh Network Alerts (MNA) to support them. We describe our experiences in deploying MNA as a real-world system to millions of users for relaying severe weather information along with the challenges faced, and the approaches for addressing them.**

***Index Terms*—peer-to-peer systems, mobile ad hoc network, delay-tolerant network**

I. Introduction

Mobile devices with programmable platforms such as android and iOS have steadily grown over the last decade, surpassing the 2 billion mark [1]. MANETs have been widely studied given their decentralized nature and potential for new applications [7], [13]. Most of the prior work on p2p networks has focused on analytical and simulation-based study of MANET behavior [8], [9], [14]. Given the outstanding practical challenges in deployoing a large number of physical nodes, real-world implementations have been limited and have not reached mass scale [6]. However, with the growth of smartphones, large-scale real-world implementations may become feasible. This paper describes a real-world implementation of a p2p delay-tolerant network, called Mesh Network Alerts (MNA), for relaying severe weather information to millions of mobile device users as part of the Weather Channel app[2] on both android and iOS platforms.

Before describing MNA, it is critical to identify applications that need p2p communication, given pervasive Internet connectivity. Doing so enables us to define key characteristics of such applications and focus on the challenges in meeting them. This paper focuses on two separate classes of applications: communication in (a) disaster-affected or remote areas and (b) congested networks in densely populated areas, e.g., sports arenas. The following are the key characteristics in these scenarios:

- CH$_0$. No communication infrastructure such as WiFi access points to fall back on
- CH$_1$. Network nodes are mobile, pattern of mobility is not predictable
- CH$_2$. New information may arrive at any time
- CH$_3$. Trustworthy information is scarce, misinformation and rumours are common place
- CH$_4$. Small payloads suffice in many cases and information retains value for a few minutes
- CH$_5$. Device battery is a scarce resource, power supply for recharging may not be available
- CH$_6$. Devices are owned by citizens, deployment of special-purpose devices is cost prohibitive

The above needs are well-recognized in the industry as well as academia with several ambitious attempts to address them, e.g., Google Loon project[3] and Facebook Aquilla[4], though with limited impact. Leveraging user mobile devices as peer nodes for a large-scale deployment has been another theme in the prior works, e.g., the Serval project [5]. Serval mesh enables p2p communication over on-device WiFi radio, but requires root access to the device via jailbreaking. Although significant leassons have been learned through these attempts, a mass-scale p2p network for such applications remains elusive.

A vast majority of the literature has focused on a traditional model of stateful, fully decentralized, reliable networking. Specifically, the nodes maintain connectivity with peers and routing is optimized with techniques based on link state or distance vectors [4], [12] focusing on optimizing the network utilization.

Given the application characteristics above, this paper identifies main practical challenges associated with modern device platforms and finds novel ways to overcome them. This leads to MNA — a new paradigm in p2p networking that employs a hybrid, delay-tolerant, and zero-routing overhead architecture. Unlike previous MANET architectures, MNA is a hybrid of a centralized and a decentralized architecture, called

[1] https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] https://weather.com/apps/ibm/meshnetworkalerts

[3] https://loon.co

[4] https://en.wikipedia.org/wiki/Facebook_Aquila

center-to-peer-to-peer (c2p2p). A central service is leveraged as the trusted source of information while the information is propagated in a decentralized fashion by peers. Such an architecture allows MNA to bypass the key issue of trust deficit in open decentralized systems and scale while retaining the ability of infrastructure-less communication in many application scenarios.

MNA is implemented on both Android and iOS as an SDK and integrated with the Weather Channel mobile apps. With extensive experiments and experience of deploying to almost 10 million users, MNA represents a way forward for large-scale p2p networks.

This paper makes the following key contributions:

- Identification of a class of applications for p2p and their key characteristics
- A deeper investigation of the practical challenges in supporting the above class of applications
- A novel c2p2p architecture implementation using multiple radio channels for modern mobile platforms (Android and iOS)
- Experimental evaluation and deployment statistics

In the following, Section II describes the practical challenges. Section III outlines the architectural details of MNA along with platform-specific implementation issues for Android and iOS in addressing the challenges. Experimental evaluation and deployment statistics are presented in Section IV. A deeper look at the literature and contrast to MNA is summarized in Section V with conclusions in Section VI.

II. Challenges

We present the main challenges for modern mobile device platforms in supporting the classes of applications described above. These have been uncovered via extensive experiments and in some cases include direct feedback from the developers of Android and iOS.

A. Operating system constraints

Due to CH_2, even when a user is not interacting with an app, or worse yet, when the device is not being used at all, the devices must continue to discover peer devices to receive and formward information. Although modern mobile operating systems such as Android and iOS offer APIs to discover and advertise information to peer devices over WiFi and Bluetooth interfaces, peer-to-peer connections do not work reliably when the same APIs are accessed while the app is in the background. Further, on Android, each WiFi p2p connection must be explicitly approved by the device user. Prior works widely document these challenges and take the approach of having special access on the devices, e.g., jail-breaking or rooting [5]. Clearly, such an approach does not scale to mass adoption.

B. Power constraints

Since devices may be offline when new weather information arrives, MNA on each peer must remain active at all times to be able to discover new information as soon as it arrives. Further, as there is no back up infrastructure (CH_0) and a set of peers in range can change at any time (CH_1), each peer is responsible for constantly forwarding available information to other peers via advertisements. However, due to CH_5, the MNA activity must keep the device battery consumption to a minimum. This is a challenge because advertising and discovery are power-hungry operations over the radio channels.

C. Testing p2p networks

Given the heterogeneity of devices owned by users and operating system distributions (CH_6), it is challenging to test whether or not MNA functions as expected on a single device. Further, running test scenarios on a p2p network at large-scale is non-trivial given that a large number of devices need to take specific coordinated action followed by coordinated observations to determine whether a test passes or fails. Further, since range and mobility affect p2p communications and they are unpredictable (CH_1), it is important to run test cases under various mobility patterns across all nodes in the network. Emulated mobility frameworks such as CORE [1] and EMANE [10] fall short as the connection latencies and wireless transmission are specific to device model and radio. This is not a challenge in traditional mobile application development as the application functionality is confined within a single connected device.

D. Trust in information

As user devices with MNA advertise on unsecure wireless protocols, it may be possible for a malicious attacker to listen for such advertisements and reverse-engineer the message formats and protocols used. Then, the attackers may generate fake messages and advertise them, e.g., a fake tornado alert. Given a lack of trusted information in such scenarios (CH_3), misinformation campaigns can have disastrous consequences. In open decentralized systems, such false messages cannot be distinguished from the real ones, and MNA will end up propagating them to as many devices as possible, “poisoning” the network. In general, veracity of such information cannot be independently verified in open decentralized systems and previous works on peer-to-peer networks do not address this challenge.

III. C2P2P System Architecture

To address (or bypass) the challenges identified above, this paper proposes c2p2p – a hybrid architecture that leverages a central service as the sole source of trusted information as depicted in Figure 1. An application-specific central service, e.g., weather.com backend, originates new information as a push message and assigns it a unique identifier. Next, the central service produces a digital signature using its private key and appends it to the message payload. Finally, header parameters specifying TTL (duration after which the message expires), peer identifier of the central service, destination peer identifier list (for unicast) or a special identifier (for broadcast),

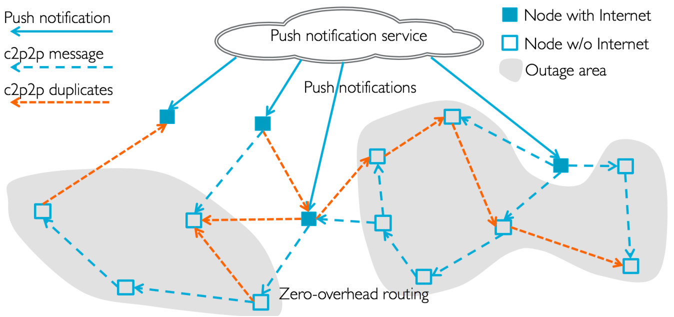


Fig. 1. c2p2p system architecture

and the time of origin are added to the message and the message is sent to a push notification service for distribution.

Mobile applications integrate MNA as an SDK including the public key of the corresponding central service. Peers having Internet connectivity receive the push messages and verify the digital signature to protect against spread of misinformation. If the message has not expired and is destined to other peers, then the receiving peers start forwarding the message to other peers. Depending on the protocol being used, such forwarding happens as a unicast or broadcast.

Since new information can arrive at any time or new peers may come into proximity, and message transmissions are not reliable, all peers store unexpired and verified messages and continue to forward them to proximal peers repeatedly. This implies that peers may receive the same message more than once but since the messages are globally unique, duplicates can be identified and ignored. A peer sending a message adds its own peer identifier to a list of forwarders appended to the message header. A receiving peer thus knows all the peers that already have the message and stops repeating the message to those peers, controlling the flooding. As there are no control messages or any other overhead for routing, we call this *zero-overhead* routing.

The following describes how the c2p2p architecture addresses the challenges identified above along with known limitations. Further, to realize this architecture on mobile devices, we evaluated a variety of protocols on both Android and iOS. As described later, WiFi DNS on Android, Bluetooth Nearby on Android, and Blutooth low energy (BTLE) on both Android and iOS are the only protocols that were found effective. Actual techniques for discovery, advertisement, and connection vary across these protocols and are described next.

A. Addressing the challenges

Operating system constraints MNA addresses these challenges via innovative techniques, without resorting to hacks that may violate user security or App-store guidelines. DNS service discovery is a widely supported protocol for all platforms [3].

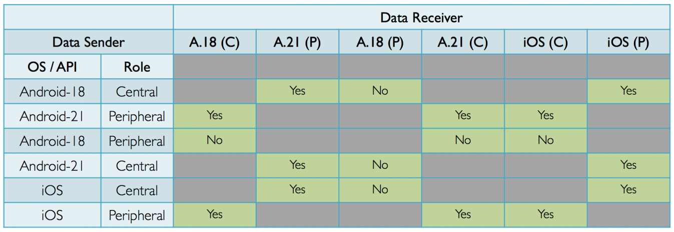
On Android, WiFi DNS improvises on this standard such that the exchange of information happens without making network connections at all. This is achieved by splitting the message payloads into small enough chunks and stuffing them into service advertisements as txt records and broadcast over multiple advertisements in a quick succession. Also, we employ foreground services to keep the discovery and advertisements active indefinitely.

On iOS, the only protocol that stays reliably active in background without violating iOS developer guidelines is BTLE. Unlike Android, iOS applications do not need to request discovery and advertisements contunually. Once a BTLE Peripheral (role that has information) is advertised and a Central (role that receives information) requests discovery, the app is allowed to go to sleep. When a matching central or a peripheral device is found, iOS wakes up the application app to handle such an event, even when the application is in the background. MNA builds on this and leverages acynchronous dispatch queues to turn each peer into a Central as well as a Peripheral simultaneously. This architecture allows bidirectional data transfers without concurrency issues.

Power constraints Due to indefinite forground services and continuous discovery and advertisement, this is primarily an issue on Android. Our approach here is two-pronged. Firstly, via extensive experiments on a large number of devices, we fine-tune the algorithms governing the intervals at which discovery and advertisements occur. In a nutshell, receiving new information during a period causes MNA to be more aggressive in discovery and advertisement. Similarly, lack of new information for a period makes the device less aggressive. Secondly, we allow "wake up" messages to be broadcast in the network ahead of an anticipated severe weather event. When devices receive such messages, they schedule themselves to remain aggressive during the specified window of time. Outside of this window, the device can afford to have long sleep cycles and conserve power. With these techniques, our testing shows less than 2% battery consumption per hour on most device models.

Testing p2p networks We developed test automation tools and processes such that multiple devices can be controlled from a single test station and follow prescribed steps to generate, send, and receives messages to play out a test scenario. Finally, the framework allows automated analysis of the observations to determine the test result. This capability was instrumental in uncovering bugs at a fast pace to meet the aggressive timeline and avoid the cost of acquisition. Further, the automation framework is general and can be expressly applied to test other apps in this fashion.

Trust in information In our approach, since the origin of weather information is the Weather Channel service, digital signatures can be attached to all weather alerts broadcast from the service. The mobile application is distributed with the corresponding public key so that digital signatures from the service can be verified. If a message fails such a verification, it is discarded and not forwarded any further.



		Data Receiver					
Data Sender		A.18 (C)	A.21 (P)	A.18 (P)	A.21 (C)	iOS (C)	iOS (P)
OS / API	Role						
Android-18	Central		Yes	No			Yes
Android-21	Peripheral	Yes			Yes	Yes	
Android-18	Peripheral	No			No	No	
Android-21	Central		Yes	No			Yes
iOS	Central		Yes	No			Yes
iOS	Peripheral	Yes			Yes	Yes	

Fig. 2. Interoperability matrix between Android and iOS devices over BTLE

TABLE I
ANDROID PROTOCOLS AND RANGE RESULTS

	Protocol	Throughput Avg. kbps	Battery per hour	Range LoS ft	iOS	Issues
WiFi	**DNS**	0.2	2%	600	No	
	WiDi	2000	3%	600	No	Needs DNS Permission
	Hotspot	2000	5%	600	No	
BT	Classic	50	2%	600	No	Unstable
	Nearby	50	2%	600	No	
	BTLE	50	2%	600	Yes	

TABLE II
IOS PROTOCOLS AND RANGE RESULTS

	Protocol	Throughput Avg. kbps	Battery per hour	Range LoS ft	Android	Issues
WiFi	Bonjour	0.2	5%	200	No	Battery
	MPC	2000	5%	200	No	Battery
	WiDi	2000	5%	200	No	Battery
	BTLE	50	2%	800	Yes	

B. Automated test framework

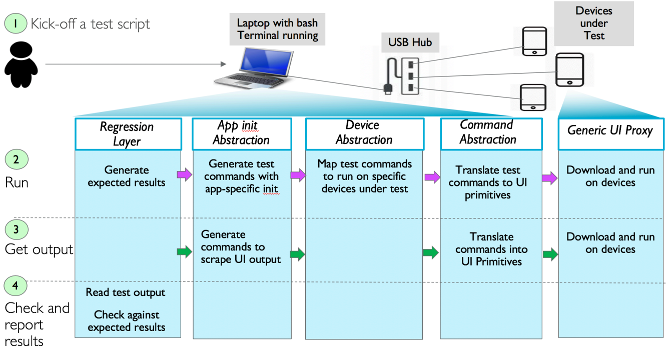

1 Kick-off a test script
Laptop with bash Terminal running
USB Hub
Devices under Test
Regression Layer
App init Abstraction
Device Abstraction
Command Abstraction
Generic UI Proxy
2 Run
Generate expected results
Generate test commands with app-specific init
Map test commands to run on specific devices under test
Translate test commands to UI primitives
Download and run on devices
3 Get output
Generate commands to scrape UI output
Translate commands into UI Primitives
Download and run on devices
4 Check and report results
Read test output
Check against expected results

Fig. 3. Automated testing system for MNA

IV. Experimental Results

A. Pilot test results

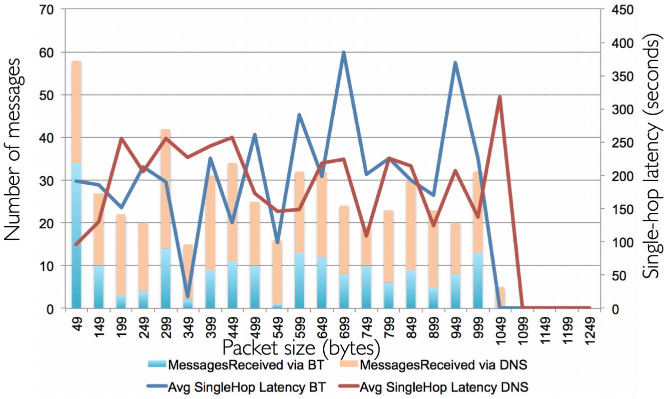


Fig. 4. Latency results from a 25-user pilot

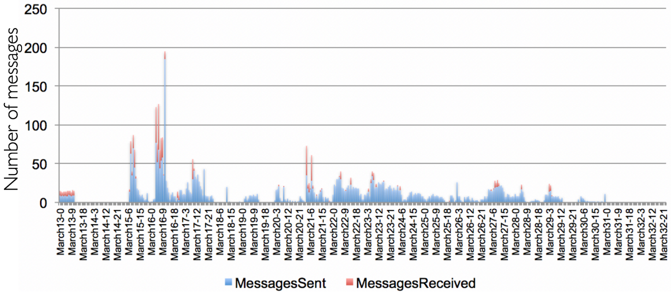


Fig. 5. MNA activity during the pilot

About 13% of the messages were received at least once (depends on density of users in the building). Users actively used the devices for 5% of the time p2p NSD achieved lower end-to-end latencies than BT Classic, perhaps due to its broadcast model and wide support across devices. Packet sizes did not affect single-hop latencies on p2p DNS, perhaps because most of the messages fit within single advert. Packet sizes did not affect single-hop latencies on BT Classic, perhaps because Bluetooth has a much higher bandwidth 2 out of 25 devices have their clocks set wrong (confusing latency results). More than 10% of the messages were received in the last 5 min of message expiry, implying messages were discarded due to expirty, even when they did not reach all devices

B. Automated test results

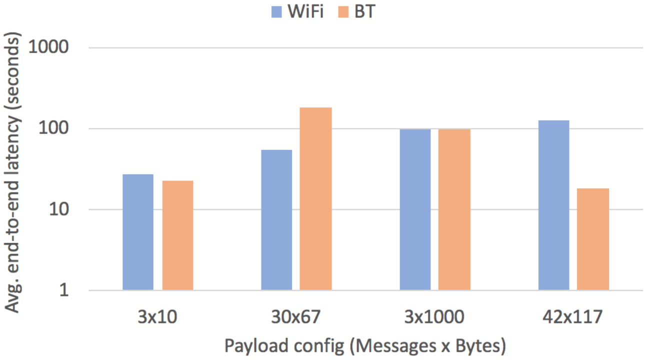


Fig. 6. Comparison of WiFi and Bluetooth interfaces in terms of end-to-end latency

V. Related Work

Apart from the state-of-the-art cited earlier, there have been other notable attempts at practical large-scale deployments and we describe them here. Before mid-2000s, most of the

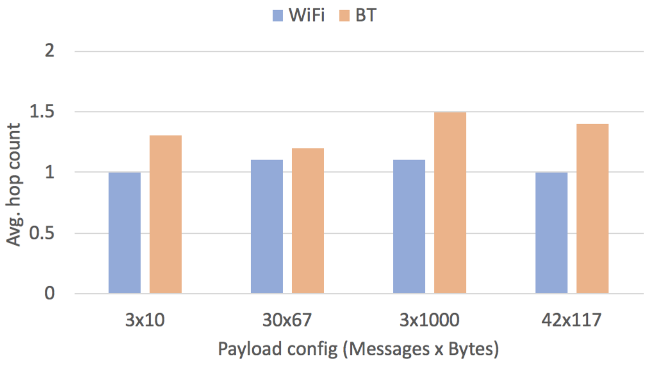


Fig. 7. Comparison of WiFi and Bluetooth interfaces in terms of network topology

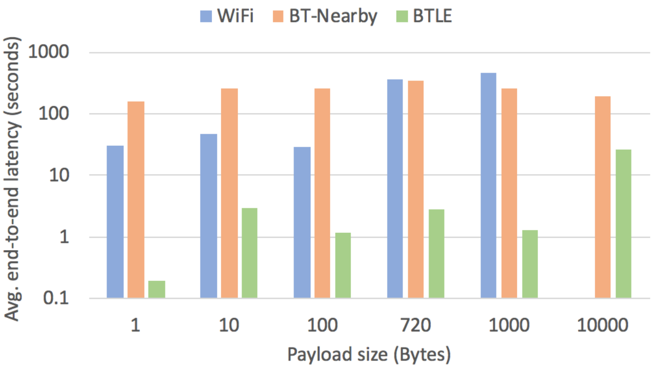


Fig. 8. Comparison of WiFi and Bluetooth Nearby interfaces in terms of end-to-end latency

research on MANETs was based on Department of Defense requirements, until commodity multi-hop ad hoc networks began to be considered [2]. However, it took another decade before wireless mesh networking was used commercially to enable smartphones to connect via Bluetooth and WiFi in a popular application called FireChat [11]. The success of FireChat, partially due to the news coverage of its use in political situations in which governments restricted access to the Internet, has led to many alternatives in the past few years. An up-to-date list of such applications is available here [5]

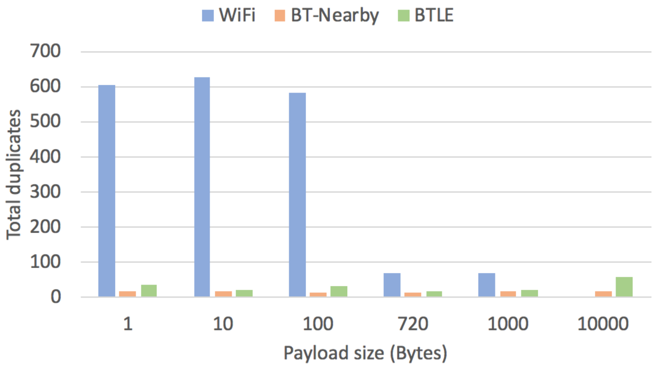


Fig. 9. Duplicity due to flooding

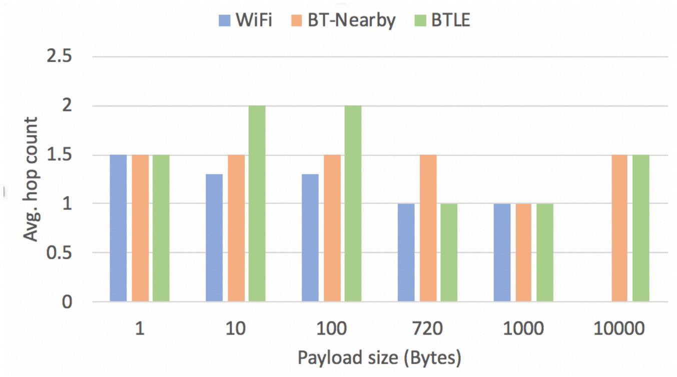


Fig. 10. Average hop count

VI. Conclusions and Future Work

Acknowledgment

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim. Core: A real-time network emulator. In *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pages 1–7, Nov 2008.
[2] R. Bruno, M. Conti, and E. Gregori. Mesh networks: commodity multihop ad hoc networks. *IEEE Communications Magazine*, 43(3):123–131, March 2005.
[3] S. Cheshire and R. Krochmal. Dns-based service discovery. *RFC 6763*, Februray 2013.
[4] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). *RFC 3626*, October 2003.
[5] P. Gardner-Stephen and S. Palaniswamy. Serval mesh software-wifi multi model management. In *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief*, ACWR '11, pages 71–77, New York, NY, USA, 2011. ACM.
[6] W. Kiess and M. Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks*, 5(3):324 – 339, 2007.
[7] J. Loo, J. L. Mauri, and J. H. Ortiz. *Mobile Ad Hoc Networks: Current Status and Future Trends*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2011.
[8] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 255–265, New York, NY, USA, 2000. ACM.
[9] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network*, 15(6):30–39, Nov 2001.
[10] Naval Research Laboratory. Extendable mobile ad-hoc network emulator (emane). Available at http://cs.itd.nrl.navy.mil/work/emane/.
[11] Open Garden Inc. https://www.opengarden.com/firechat/.

[5] https://alternativeto.net/software/firechat-by-open-garden/

[12] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. *RFC 3561*, July 2003.

[13] C. E. Perkins. In *Ad Hoc Networking*, chapter Ad Hoc Networking: An Introduction, pages 1–28. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[14] X. M. Zhang, Y. Zhang, F. Yan, and A. V. Vasilakos. Interference-based topology control algorithm for delay-constrained mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 14(4):742–754, April 2015.