A Smartphone-based Network Architecture for Post-disaster Operations Using WiFi Tethering

AMITANGSHU PAL, Temple University
MAYANK RAJ, IBM
KRISHNA KANT, Temple University
SAJAL K. DAS, Missouri University of Science and Technology

Electronic communication is crucial for monitoring the rescue-relief operations and providing assistance to the affected people during and after disasters. Given the ubiquity of smartphones, we envision that smartphones with lost connection (due to damage) to the communications infrastructure are nevertheless integrated seamlessly into the network as far as possible. To achieve this, we propose to build ad hoc subnetworks of disconnected smartphones using the WiFi tethering technology and ultimately connect them to either the emergency communication equipment deployed in the disaster area or to other smartphones that have still the network connectivity. The proposed architecture for such integration and a defined software-based control through the emergency control center (ECC) enables battery aware collection of critical data through smartphone sensors. The developed solution supports mobility of all smartphones, including those that have lost direct cellular connectivity as well as those that have not and are willing to act as gateways. We demonstrate how the proposed scheme can be tied to the standardized wireless emergency alert service and how it can effectively handle mobility tolerant device discovery and data transfer.

CCS Concepts: • **Networks** → **Network design principles**; **Network protocol design**; **Network algorithms**; **Mobile networks**; **Mobile ad hoc networks**;

Additional Key Words and Phrases: Disaster communications, WiFi tethering, ad-hoc network, grid-based quorum

ACM Reference format:
Amitangshu Pal, Mayank Raj, Krishna Kant, and Sajal K. Das. 2020. A Smartphone-based Network Architecture for Post-disaster Operations Using WiFi Tethering. *ACM Trans. Internet Technol.* 20, 1, Article 6 (February 2020), 27 pages.
https://doi.org/10.1145/3372145

6

S. K. Das is also an International Visiting Professor and SERB-sponsored VAJRA faculty at IIT Kharagpur, India; and Satish Dhawan Visiting Chair Professor at IISc Bangalore. This research was partially supported by NSF Grants No. CNS-1461932, No. CNS-1818942, No. CCF-1725755, No. CNS-1545037, No. CNS-1545050, and No. DGE-1433659.
Authors' addresses: A. Pal (corresponding author) and K. Kant, Temple University, 1925 N. 12th Street, Philadelphia, PA, 19121, USA; emails: {amitangshu.pal, kkant}@temple.edu; M. Raj, IBM, 11550 Ash Street, Leawood, KS, 66211, USA; email: mraj@us.ibm.com; S. Das, Missouri University of Science and Technology, 500 W. 15th Street, Rolla, MO 65409, USA; email: sdas@mst.edu.

1533-5399/2020/02-ART6 $15.00
https://doi.org/10.1145/3372145

1 INTRODUCTION

Of late, there has been an increased number of large-scale natural disasters extending over several days or even weeks. Numerous example abound worldwide, including the Nepal earthquake (2015), Hurricane Maria in Puerto Rico and Hurricane Harvey in Houston (2017), as well as recent cyclone Fani in India and devastating floods in Southeast Asia (2019). Many of these disaster scenarios dynamically evolve, often in unexpected ways. Thus, situation awareness of their evolution is crucial for providing proper emergency response services in terms of rescue and evacuation, health care, shelter, food, medicine, and other relief operations. The widespread use of smartphone technology has increasingly made smartphones the lifeline for communication, people tracking, and need assessment, particularly during and in the aftermath of disasters. Given that smartphones carry a multitude of sensors (e.g., camera, microphone, kinetic/vibration sensors, GPS, etc.), the data captured by them can be invaluable in better assessing the damage and needs in disaster-affected areas. In other words, beyond typical manual use (such as voice calls, posting text/photos, or videos to the social media), smartphones can also provide enormous additional value through (semi)-automated data collection, analysis and decision-making during or aftermath of disasters [1–4]. This article aims to explore a variety of such issues in automated data-driven usage of smartphones during disasters, particularly when the existing communication networks may be partially or fully damaged.

To support data-driven disaster response services, we assume the existence of a remotely located operational Emergency Command Center (ECC), which forms the central hub for data collection, analysis, and decision making. We also assume that the ECC has internet access and its operation is not disrupted by the disaster. The normal way for smartphones to reach the ECC is via cellular or WiFi connections to local access points. However, they may experience scattered outage requiring establishment of ad hoc networking infrastructure between smartphones. In this article, we explore a scheme, called *E-DARWIN* (*Energy Aware Disaster Response Network using WiFi Tethering*), to build an ad hoc network in pockets of the disaster area that lost (or has very poor) cellular connectivity. E-DARWIN exploits the WiFi tethering capability available in most of today's smartphones and is activated via an App that is assumed to be installed on all willing smartphones ahead of time. Each of the E-DARWIN ad hoc networks would ultimately need to connect to a *gateway device* having access to the Internet and eventually to the ECC. In turn, the ECC directs the data collection to ensure the most effective use of smartphone sensors, their remaining battery power, and latency-throughput characteristics of this network.

The gateway function can be provided by a variety of devices. For example, satellite phones carried by the rescue personnel, or satellite/cellular access points (APs) air-dropped or deployed via emergency communication vehicles (ECVs) could act as gateways. We assume that the gateway devices also have WiFi capability through which the non-gateway devices may connect. Even ordinary smartphones could serve as gateways if the disaster does not wipe out the entire network. A more common scenario is scattered pockets (islands) that either have no or very poor cellular connectivity owing to the actual damage or extreme network congestion. The proposed ad hoc network would span such pockets and ultimately connect to the smartphones at the periphery of the pocket that do have cellular availability. In fact, in the initial stages of the disaster, such connectivity is perhaps all that is available. The network will evolve due to the movement of people and deployment of additional emergency assets. Thus, the evolution is an important component of the mechanisms discussed here.

The key reason behind our focus on the (semi-)automated network build-out is that the affected people carrying smartphones may be incapacitated, confused, or otherwise not in the best position to manage network formation or data transfer manually. Although the network can carry user

generated content (e.g., text messages or photos), its limited bandwidth and significant delays are not suitable for real-time or heavy-duty content (e.g., voice or video). User generated texts and low-resolution photos can be carried, but require a careful allocation of capacity between the ECC directed and user directed usage, which is outside the scope of this article. It is also possible to trigger network build-out based on the embedded sensors detecting relevant signals themselves. Two examples of the latter are (a) seismic vibrations-based earthquake detection [5–7], and (b) monitoring appropriate emergency sirens [8, 9]. Such enhancements can be easily incorporated into our proposed scheme.

Despite the simplicity of the concept, building a useful ad hoc network involves many challenges. Addressing such challenges constitutes the main contribution of this article, as summarized below.

(1) Determine when and how to build the ad hoc network, since a loss of connectivity alone is inadequate, thus unnecessary build-out must be avoided to conserve smartphone batteries.
(2) Efficiently build the network given the challenges of mobility, requirement of complementary mode (client vs. hotspot) for two devices to connect via WiFi tethering, and the need to find and connect to the gateway devices having network connectivity.
(3) Efficiently schedule data transfer between smartphones given the dynamics, low throughput, and high latency of the ad hoc network.

To the best of our knowledge, ours is the first work that investigates the construction of ad hoc network for disaster scenarios using WiFi tethering technology and exploiting the proposed data routing scheme.

The rest of the article is organized as follows. Section 2 summarizes the related works while Section 3 describes the background and proposed architecture of the E-DARWIN framework. Section 4 discusses network initialization and Section 5 presents the analytic model for assessing its performance. Section 6 introduces the grid quorum-based communication mechanism and shows how it can be extended to WiFi tethering. Section 7 describes how the grid quorum mechanism can be used for energy efficient data forwarding. Section 8 presents experimental evaluation of the E-DARWIN architecture with the help of a prototype implemented on the Android platform, as well as large-scale simulation study. Finally, Section 9 concludes the article.

2 RELATED WORK

Using special purpose sensors for constructing disaster monitoring network and capturing data has been extensively explored in the literature [10–13]. The use of wireless devices like smartphones, PDAs and tablets widely available among people in the disaster affected region also offers a practical approach [14, 15]. Even though existing solutions utilize the ad hoc mode of operation in wireless devices to construct a disaster recovery network [16–19], implementing them on current wireless devices requires modifications of the transceiver driver firmware, root access to the kernel, or jailbroken devices [20, 21]. This is undesirable, because such solutions cannot be seamlessly supported across all devices, and are also illegal in many countries. This motivated us to explore the use of technologies readily available on wireless devices (e.g., WiFi Direct and WiFi Tethering) for constructing the disaster recovery network.

WiFi Direct or WiFi Peer-to-Peer (P2P) is a standard, which allows the devices to communicate with one another without the need of a wireless access point (AP) [22]. WiFi Direct works by embedding limited wireless APs in the devices and the capability to "pair" two such devices. Although WiFi Direct-based ad hoc networking for disaster communications has been explored in Reference [23], this capability is not available in all mobile platforms. Therefore, for purposes

of this article, we selected WiFi Tethering (or WiFi-T) as the preferred technology noting that the proposed solution applies to other technologies as well.

In References [24–27], the authors studied energy consumption of a device using WiFi-T, and proposed mechanisms to improve the device's energy efficiency. Having established the viability of the use of mobile devices, the authors explored how to communicate with other devices in an opportunistic network environment [28] and create computing cluster grids [29]. A role-based architecture is adopted in such solutions wherein the devices take up the role of hotspot or client to set up a mobile ad hoc network [30, 31]. In contrast, the authors in Reference [31] proposed to use network virtualization to let the devices assume simultaneously the roles of hotspot and client. However, virtualization of the network interfaces is not supported in most of the current mobile devices. Moreover, the authors in References [30, 31] did not address issues related to network formation, routing, and maintenance.

Opportunistic networks based on occasional encounters with clients acting as access points (e.g., police patrol car with a special emergency AP) have been considered in the literature [32–34]. However, our goal is to build networks that are not ephemeral, as they are more likely to be useful for persistent monitoring.

Finally, we remark that a preliminary version of this work with a different data forwarding scheme was published in a conference [35]. In contrast, the data forwarding scheme considered in this journal version is based on the grid quorum technique, and designed to handle smartphone mobility. Additionally, we have proposed in details the network initialization procedure by exploiting the Wireless Emergency Alert (WEA) service widely deployed in the US and available on nearly all smartphones. While the conference version presented a game theoretic mechanism to exploit data redundancy such that the data transmissions (hence battery drainage) can be reduced, the journal version does not focus on the data redundancy aspect.

3 BACKGROUND AND PROBLEM DESCRIPTION

3.1 Network Overview and Key Assumptions

As discussed above, every smartphone is assumed to be equipped with E-DARWIN app, which can help establish the ad hoc network to collect data under program control of the available sensors in smartphones (e.g., camera, microphone, gyroscope, GPS, etc.), and transferring the data to the next level of the network. The latter consists of *gateway devices*, including smartphones with surviving/established cellular links and emergency APs. We assume that a few surviving base stations and "Cell On Wheels" (COWs) are operational in their limited coverage area. The COWs are mobile, portable cell towers and transceivers mounted on trailers or trucks for easy deployment in the affected areas [36]. The emergency WiFi APs are deployed in conjunction with the satellite gateways [37] in the affected region. Each satellite gateway is composed of a very-small-aperture-terminal (VSAT) dish antenna, and a satellite modem that can be easily assembled and disassembled for portability [38]. When the road connectivity is available, the WiFi APs with satellite gateways are deployed at various locations in the affected region with the help of emergency vehicles. If no road connectivity is available in the affected region, then the equipment can be carried or air-dropped by emergency crews as proposed in Reference [38], and deployed similarly.

Since the APs are intended exclusively for assisting with the emergency network, we assume that they are pre-authenticated by the ECC and connected to the ECC using traditional technologies, such as satellite network [37]. In addition, all smartphones are assumed to be capable of reaching the ECC via a well known URL. The ECC is established by a public aid agency, like the Federal Emergency Management Agency (FEMA) in the US, in some secure location for providing disaster relief; it is a permanent entity that only needs to be fired up whenever necessary. We

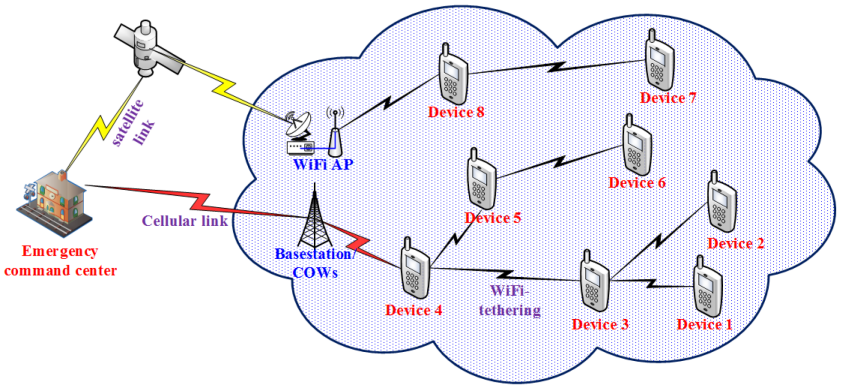


Fig. 1. Network architecture of E-DARWIN.

further assume that the ECC has the necessary authority, can be trusted, and goes online shortly after a disaster event. The ECC forms the apex of the entire emergency network, determines data collection needs, collects and analyzes all data, instructs the network reconfiguration, and so on.

The E-DARWIN architecture is illustrated in Figure 1. Although there are many important issues to address at the next level network (e.g., positioning of emergency communication vehicles or ECVs, choice of communication technologies, etc.), our focus in this article is mostly on the smartphone end of the disaster network.

3.2 Possible Wireless Technologies in Disaster Environment

Although there exist many technologies (currently available or under development) that can support P2P communications between smartphones, most of them suffer from some limitations. For example, the long term evolution (LTE)—the dominant 4G technology—is also being extended for P2P communications with emerging standard of LTE direct [39]. However, LTE direct does require an intermediary (cell tower) for device discovery and connection establishment, which is problematic in disaster scenarios. Nevertheless, with the deployment of microcell/picocell towers, the LTE and LTE direct can be extremely useful in disasters. Deploying such services may require hours and days, however, this is not the focus of this article.

Similarly, although the ad hoc mode WiFi is a rather old technology, it is still not widely supported and often requires root access on the device to set up [28]. Two other promising technologies are WiFi Direct and WiFi Tethering. Indeed, WiFi Direct or WiFi P2P is a newer standard that allows device to device communication by embedding limited wireless AP in the devices. Using WiFi Direct, a few devices can form a group such that one of them acts as a group owner, while the others as group members. According to the WiFi alliance [40], "*A WiFi Direct-certified network can be one-to-one, or one-to-many ... Connection to multiple other devices is an optional feature that will not be supported in all WiFi Direct-certified devices; some devices will only make 1:1 connections.*" Thus, WiFi direct devices that support only 1:1 connection are not quite suitable for a multi-device network.

To this end, WiFi tethering (in short, WiFi-T) comes out as a prominent technology that allows sharing of internet connection of a mobile device with other nearby devices. WiFi-T is ubiquitously supported by all major device manufacturers and available across all major mobile OS platforms, such as iOS 4.3+, Android 2.2+, and Windows 7.5+. WiFi-T also allows a device to act as a WiFi hotspot while other devices in its communication range can connect to it. The use of WiFi-T requires a device to alternate between the client and hotspot modes to forward data (since a link can

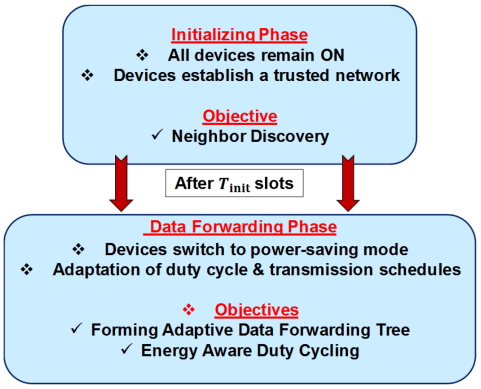


Fig. 2. Two phases of the E-DARWIN Framework.

exist only between a client and a hotspot). This essentially results in a delay toerant network (DTN) with most links established selectively and intermittently as needed, which is also crucial from the perspective of minimizing drainage of smartphone batteries. The key challenges include mutual device discovery or initialization, building the entire network with ECC at its apex, efficiently collecting data, and delivering them to the ECC by taking into consideration the smartphone's battery drainage, available bandwidth, reachability, data transfer latency requirements, and redundancy exploitation in the collected data to reduce battery drainage.

Note that nearly all current smartphones support (in the infrastructure mode) WiFi standard; the WiFi-T capability is also available in nearly all existing smartphones; and a WiFi hotspot can connect to multiple clients. Hence, we focus on WiFi-T as the key communications technology for building ad hoc networks in the wake of disasters. While WiFi direct is also becoming popular, its penetration is still much lower than WiFi-T. Although it is possible to integrate WiFi direct into E-DARWIN, it is beyond the scope of this article.

3.3 WiFi Tethering-based Networking

As illustrated in Figure 2, the smartphone network in the proposed E-DARWIN framework operates in two phases—*Initialization phase* and *Data Forwarding phase*. We assume that all devices have installed and enabled the E-DARWIN app. When triggered via an emergency, the app on each device goes into the Initialization phase to discover neighbors and build an ad hoc network; ultimately, it either confirms the emergency and goes into the Data Forwarding phase or simply folds, i.e., no real emergency. In the Data Forwarding phase, each device adapts its duty cycle and transmission schedules based on its available battery capacity and data transmission needs, and periodically forwards the sampled data. The two phases are detailed in Sections 4 and 7.

The WiFi-T-based operations require each device in one of the following three states:

Dormant: By default a device stays in the dormant state to conserve energy, until it is scheduled to act either as a WiFi hotspot or client. The Dormant state disables all network interfaces of the device, which runs in a low-power mode with CPU sleeping. This state is different from being dormant in the medium access control (MAC) layer, in which an active data connection exists and only the wireless interface is periodically switched off to conserve energy.

WiFi Hotspot: In this state, the device repeatedly broadcasts network discovery beacons such that any neighboring clients can associate with it and exchange data packets. Devices act as hotspot for a predetermined time interval before entering the dormant or client state.

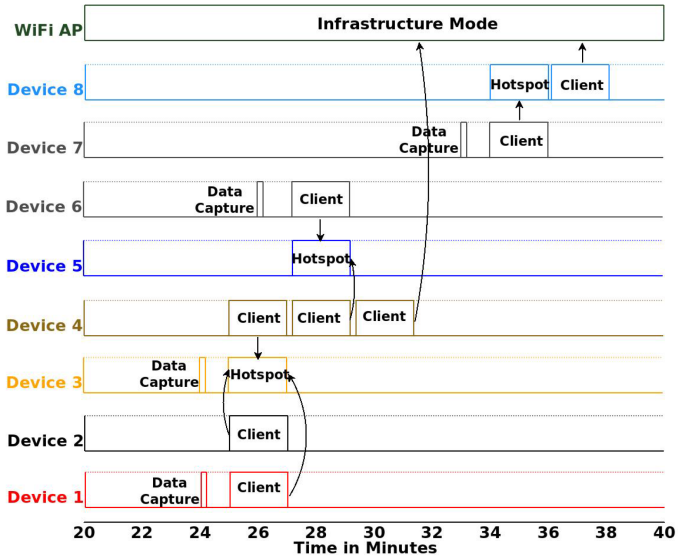


Fig. 3. Functioning of E-DARWIN.

WiFi Client: In this state, the device periodically scans the wireless channel for advertisements from the hotspot and associates with it. On successful association, the client devices connected to a hotspot participate in data forwarding or receiving. A device stays in the client state as long as there are some data transfer with any of its hotspot neighbors, and accordingly enters the dormant or hotspot state.

A device can capture data while in the hotspot or client state. If a device is scheduled to capture data in the dormant state, then it will wake up to do so and then become dormant again.

3.4 An Illustrative Example

Figure 3 illustrates typical behavior of the devices in E-DARWIN following the initialization phase. Let D_m represent the device $m \in \{1, 2, \ldots, 8\}$. Assume that at $t = 25$ mins, D_3 is scheduled to act as a WiFi hotspot while D_1, D_2, and D_4, which are in the communication range of D_3, wake up as WiFi client and associate with it. Thus, at $t = 25$ mins, D_1 and D_2 can forward their stored data packets to D_3, which in turn can forward to D_4 as soon as it receives them along with its own data packets. Assuming that the devices are configured to act as a WiFi hotspot or client for 2 mins, the devices stay in their respective states till $t = 27$ mins before becoming dormant.

At $t = 27$ mins, D_5 and D_6 are scheduled to wake up in the hotspot and client modes, respectively. D_4 goes into client mode and associate with D_5. Thus, D_6 sends its packets to D_5, which is forwarded to D_4 in the same time slot.

However, as D_4 has stored packets to forward and it is in the range of WiFi AP, it switches to client mode at $t = 29$ mins and associates to the AP to forward its packets. D_4 also forwards the packets sent by other devices to the AP in the same slot. Similarly, at $t = 34$ mins, D_8 goes into hotspot mode whereas D_7 is scheduled to wake up in client mode. Thus, D_7 associates with D_8 at $t = 33$ mins to forward its packets, which is then forwarded by D_8 to the WiFi AP in the next time slot at $t = 36$ mins. In this fashion, the devices autonomously form a multi-hop tree topology to forward their sensed packets to the APs.

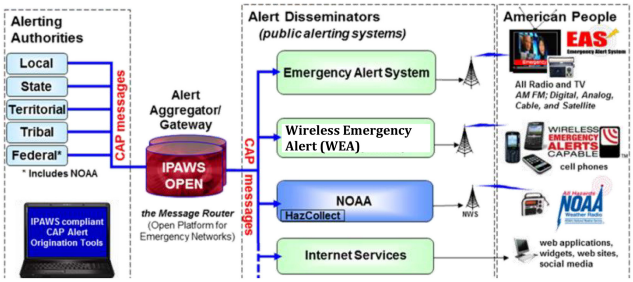


Fig. 4. Integrated Public Alert and Warning System. Image taken with permission from FEMA [41].

4 INITIALIZATION PHASE

4.1 Activation of E-DARWIN Application

The initialization of E-DARWIN network aims to address two interdependent problems: (a) based on WiFi tethering, discover neighbors to build locally connected clusters of smartphones, which can connect to the Internet via gateway devices; and (b) with minimum energy expenditure, ensure that the network can serve emergency (disaster) scenarios.

For this purpose, particularly in the U.S. context, we link E-DARWIN to official emergency alert systems, such as the Wireless Emergency Alert (WEA) system. Figure 4 depicts current systems in the US that use the Common Alerting Protocol (CAP) to broadcast alerts from official sources via the open-source Integrated Public Alert and Warning System (IPAWS) with a goal to integrate various subsystems including WEA involving wireless cellular carriers [42]. (Similar systems likely exist elsewhere in the world.) Currently, the majority of cellular service providers in the US implement WEA supported in all Android/IOS phones. Note that WEA messages ride on the control channel and thus are not affected by the network congestion.

We assume that a WEA alert, such as imminent threats to safety (e.g., tsunami, tornado, hurricane, typhoon warnings) triggers the E-DARWIN app. This is a reasonable assumption, because WEA alerts are broadcast to the affected areas; after receiving the WEA alert message, a phone parses and analyzes to determine if it is *vulnerable*. For simplicity, we assume that non-vulnerable phones do not have any role to play unless the app is started manually and instructed to participate. The significance of this assumption is that we do not expect non-vulnerable phones to participate in the network (except by deliberate choice of the user who starts the app manually).

4.2 Starting Network Buildout

Even after the E-DARWIN app determines a phone is vulnerable, it may not start building the WiFi-based network until it actually loses the cellular connection. In other words, a phone having cellular connection will not join in and act as a gateway. Thus, different procedures are needed for *Affected Vulnerable Devices* (AVDs) that actually lose cellular connection, and *Unaffected Vulnerable Devices* (UVDs) without losing cellular connection. Such devices can use their International Mobile Equipment Identity (IMEI) or other unique information as ID in the discovery process; it is reasonable to assume that each device maintains fairly accurate time (down to a few seconds). A vulnerable device losing cellular connection continuously for some fixed time (say, 15 minutes) classifies itself as AVD and starts the discovery process. If it later regains cellular connectivity for some fixed time, then it can reclassify itself as UVD. We normally expect AVD to UVD transition to be very infrequent, hence a completely separate mechanism for UVDs is discussed later. The process of identifying a disaster and switching from the normal mode to the vulnerable mode is depicted in Figure 5.

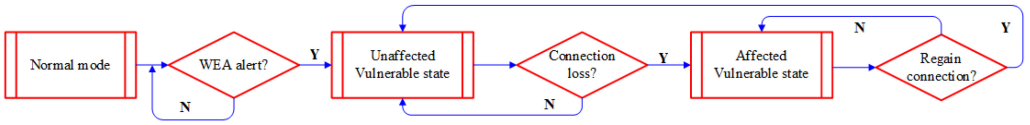


Fig. 5. Flow diagram of E-DARWIN app initialization, discovering a disaster, and switching from normal mode to vulnerable modes.

As the discovery proceeds among the AVDs and the WiFi tethering network expands, the discovery process ultimately includes phones having cellular access and emergency APs deployed in the disaster area. In reality, the deployment of emergency APs is likely to take some time following the loss or degradation of the cellular network. Hence it may be necessary to tear-down an already built cluster of nodes that cannot reach the ECC. Yet, it should be possible to retry when new emergency APs are eventually deployed or cellular access becomes at least partially functional.

Because of gradual impact of most disasters, it is possible that initially there are enough UVDs in an area such that every AVD cluster can use one or more of them as a gateway. Later on, as the cellular connectivity decreases, some AVD clusters may become entirely isolated for some time, and later reach the ECC again through APs as they become online. In any case, a full coverage is not possible in general, and our aim is to use the AVDs that can reach the ECC for data transfer while continuing to discover potential new pathways. The details on the discovery process is described in Section 4.3. The selection of UVDs as gateways is controlled by the ECC that recruits them depending on their density in a given area, the number of APs in the area, traffic needs, and so on. However, the details of such recruitment process is beyond the scope of this article.

4.3 Neighbor Discovery

When the E-DARWIN app is activated, the devices need to discover their neighbors including their schedule. As discussed, there are two types of phones to be considered during a disaster: AVDs and UVDs. (We assume that an AVD cannot access the Internet even if it was connected to a normal WiFi AP before the interruption.)

Let us first describe how the AVDs discover each other. We assume that the time is divided into "slots" with a duration of Δ. Existing solutions in the literature for neighbor discovery rely on the coordination between the devices to ensure that their radios are simultaneously turned on for discovering one another [43, 44]. With WiFi tethering, they must have WiFi radios turned on and remain in different states (e.g., one acting as WiFi hotspot and the other as WiFi client). To address this problem, the devices are restricted to stay in the hotspot state during the initialization phase for T_{init} slots. During this phase, the devices consecutively take up the role of WiFi client in the next C_{next} time slot chosen uniformly at random from the next $[1, C_{\text{disc}}]$ time slots. This allows the devices to be in different states after random time slots, and thus discover one another.

During the initialization phase, a device in the hotspot state waits for one of its neighbors to become client and associate with it. When the device becomes the client, it stays in that state for one slot time. The client device scans the wireless channel for advertisements from one of its neighbors, which may be in hotspot state. On receiving the advertisement from the hotspot, the client device associates with it. On successful association, the devices add each other to the neighbor list and exchange each other's schedule, which is described in Section 6.

The UVDs also need to take part in the discovery process so they can connect to AVDs, but it is unreasonable to expect them to continuously try to discover their neighbors. From the perspective of E-DARWIN, a UVD will turn on its WiFi radio in the *client mode* for short periods so that it can discover any nearby AVDs in hotspot mode. The wake up frequency can be either periodic or

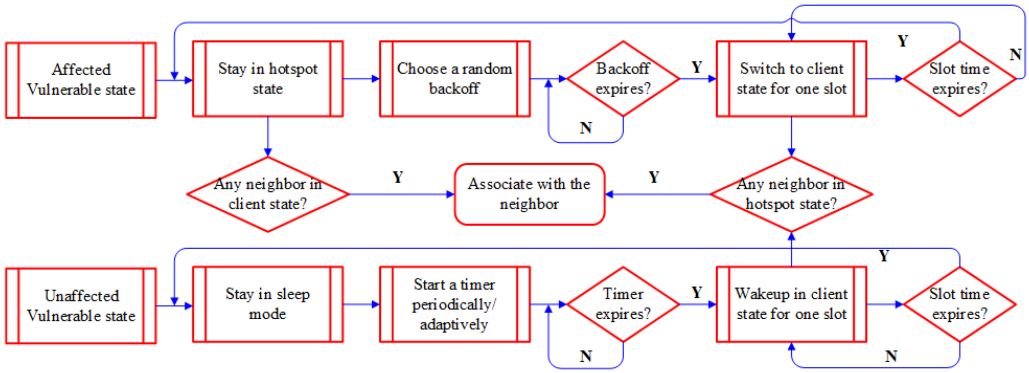


Fig. 6. Flow diagram for discovering the neighboring devices in the Initialization phase.

adaptive. In the adaptive wake up mechanism, a UVD starts with a small sleep duration (say, 15 minutes) and as long as no new AVDs are found, it is increased exponentially for the next round up to a maximum threshold $\mathcal{I}$ (say, 1 hour). If any new AVDs are found, then the sleep duration is reduced to the starting value. Such a discovery mechanism ensures that if there are no AVDs in the neighborhood of a cluster of UVDs, they will not form any network, because all of them will be in the client mode. At the same time, any AVDs will be contacted and thus will be able to use the UVD as a gateway. Each device also keeps track of who it has talked to recently (for few hours) to avoid repeated connection establishment and authentication message exchange. The discovery process is shown in Figure 6.

The discovery mechanism has an impact on a UVD if the latter accesses the Internet by connecting to a WiFi access point (AP). In this case, the UVD will be required to periodically break its association with the normal AP so that it can connect to the AVDs. This disrupts the user experience of the UVD unless the cellular connection can be transparently substituted for WiFi, while the UVD works as a gateway to the AVDs. Most smartphones can switch back and forth between cellular and WiFi accesses without user intervention. Furthermore, the switching needs to be better coordinated, since it is triggered by a planned action rather than, say, weak signal. Finally, as the UVD switches back to connect to the WiFi AP, it should associate with its normal AP instead of one of the AVDs that happen to be in the hotspot mode.

5 ANALYTICAL MODEL FOR DISCOVERY PERFORMANCE

This section presents an analytical model that studies the impact of the algorithm parameters on the probability of an AVD discovering its neighbors. The necessary parameters are listed in Table 1. As mentioned in Section 4.3, we divide the time into discrete "slots" with duration Δ chosen as the minimum time required by a device to scan the channel for advertisements from the hotspot, receive them, and associate with the hotspot.

5.1 Probability of AVD Discovery

Let A and B denote two devices in the initialization phase. Each device chooses a random time slot C_{next} to become a client from the next C_{disc} time slots. Given each device remains in the initialization phase for T_{init} time slots, they will become client at least $\rho = \lfloor T_{\text{init}}/C_{\text{disc}} \rfloor$ times. For simplicity, we assume that T_{init} is a multiple of C_{disc} such that ρ is an integer.

Now A and B will discover each other if they do not become client in the same time slot. Thus,

$$P(A \text{ and } B \text{ will discover one another}) = 1 - 1/(C_{\text{disc}})^{\rho}. \quad (1)$$

Table 1. Table of Notations and Their Meanings

Δ	Duration of a slot time during the discovery phase
T_{init}	Number of time slots in the initialization phase
C_{disc}	Maximum number of time slots for which AVD may stay in hotspot mode during initialization phase
C_{next}	Random time slots $\in [1, C_{\text{disc}}]$ after which an AVD goes to client mode during initialization
ρ	Number of times an AVD goes on client state during initialization phase
$\mathcal{I}$	Maximum sleep duration of a UVD
s	Maximum sleep duration of a UVD in the number of slots, i.e., $s = \mathcal{I}/\Delta$
$\mathcal{N}$	Number of UVD neighbors of an AVD
$(\mathbb{R}_a, \mathbb{C}_a)$	Row and column chosen by device a

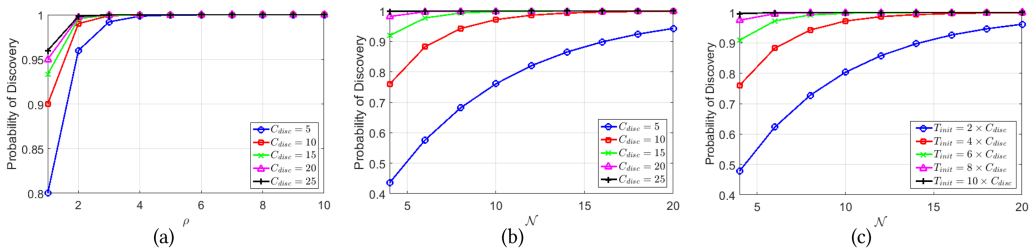


Fig. 7. (a) Probability of discovery between two AVDs with varying C_{disc} and ρ. Probability of discovery across UVDs and ADVs B with (b) varying C_{disc} and (c) varying T_{init}.

Figure 7(a) shows the probability that two AVDs discover each other with different C_{disc} and ρ. We can observe that as C_{disc} increases, the probability that the two devices may select the same time slot to become client, also decreases, thus improving the probability of device discovery. We can also observe that the probability of device discovery increases with increasing ρ. From Figure 7(a), we notice that AVDs need to become client at least 4 times to discover one another with >99% probability, which is ensured if $T_{\text{init}} \geq 4 \times C_{\text{disc}}$.

5.2 Probability of Discovery Across UVDs and AVDs

We now derive the probability that an AVD is discovered by at least one of its $\mathcal{N}$ UVD neighbors, during the initialization phase. We consider the worst case scenario where the sleep interval of the UVD is $\mathcal{I}$ time units, which is equivalent to $s = \mathcal{I}/\Delta$ slots.

Let us consider two neighbors A and B representing UVD and AVD, respectively. In the initialization phase, an AVD remains in the hotspot mode in any time slot and occasionally goes to the client mode with a probability of $1/C_{\text{disc}}$. Thus, at any time slot during the initialization phase, the AVD B remains in the hotspot mode with probability $P_h = (1 - 1/C_{\text{disc}})$.

The probability that A wakes up within the initialization phase of B is illustrated Figure 8. Assume that at time instance 0, B starts its initialization phase with $T_{\text{init}} = 5$ slots. In the worst case, the UVD A wakes up once in every s slots, which is assumed to be 8 in Figure 8. Notice that at any time instance, a UVD stays in any one of the s states defined as the number of slots after which A will wake up. For example, in Figure 8, we assume $s = 8$. That is, at time instance 0, if A is in state $S0$, then it wakes up at slot 0; whereas, if it is in $S7$, then it wakes up after 7 slots. Thus, at time instance 0, a UVD A stays in any one of the s states with probability of $1/s$. Then, A wakes up within the initialization phase of B if it stays in any one of the states $\{S0, S1, S2, S3, S4\}$ at time instance 0. In other words, A wakes up in any one of the T_{init} slots with a probability of $\min(1, T_{\text{init}}/s)$. Thus,

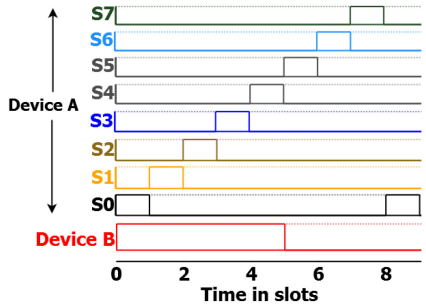


Fig. 8. An illustrative example.

the probability that a UVD discovers its neighboring AVD is given by

$$P(A \text{ and } B \text{ in client and hotspot mode}) = \begin{cases} P_h T_{\text{init}}/s & \text{if } P_h T_{\text{init}} < s, \\ 1 & \text{Otherwise.} \end{cases} \tag{2}$$

If there are $\mathcal{N}$ UVDs in the vicinity of B, then B will be discovered by at least one of its UVD neighbors with a probability:

$$P(B \text{ is discovered by at least one UVD}) = \begin{cases} 1 - (1 - P_h T_{\text{init}}/s)^{\mathcal{N}} & \text{if } P_h T_{\text{init}} < s, \\ 1 & \text{Otherwise.} \end{cases} \tag{3}$$

We assume $\Delta = 30$ seconds, and $\mathcal{I} = 1$ hour, which is equivalent to $s = 120$ slots. We keep $T_{\text{init}} = 4 \times C_{\text{disc}}$, and vary C_{disc}. From Figure 7(b), observe that as C_{disc} increases, T_{init} also increases, which improves the probability of discovery in between UVDs and AVDs. When $C_{\text{disc}} = 25$, T_{init} becomes 100, which makes the probability of discovery almost equal to 1 as observed from Equation (3).

We next assume $C_{\text{disc}} = 10$ and vary T_{init}, as shown in Figure 7(c). Similar to Figure 7(b), the probability of discovery increases with increasing T_{init}, due to the longer discovery period. We also notice that the probability of discovery increases with the increase in the number of UVD neighbors. From Figures 7(b) and 7(c), we conclude that as far as $T_{\text{init}} \geq 4 \times C_{\text{disc}}$, $C_{\text{disc}} \geq 10$ and $\mathcal{N} \geq 10$, the probability that the AVDs are discovered by their UVD neighbors is >95%. Also, with $\mathcal{I} = 1$ hour, the UVDs need to keep their WiFi radio on for only 0.833% of their time, which ensures that the scheme does not put a significant burden on the UVDs in the discovery phase.

6 QUORUM GRID-BASED COMMUNICATION

After T_{init} slots the AVDs move to the data forwarding phase. The UVDs also move to this phase T_{init} slots after discovering their first AVD. In the data forwarding phase (also called adaptation phase), the devices mostly remain dormant to conserve their battery power, and occasionally go to the hotspot (or client) mode for necessary communications. For this phase, we adopt a *quorum*-based duty-cycling scheme for the devices as described below. Assuming that the time is divided into *slots*, each being large enough to accommodate multiple transmissions, we now define the quorum set as follows:

Definition 6.1 (Quorum Set). A quorum set represents the time slots when a device wakes up. In the non-quorum time slots, devices are allowed to enter the sleep mode to conserve energy.

The concept of quorum set was initially proposed in the context of energy-aware ad-hoc networks [45] and low-power sensor networks [46, 47]. To ensure that the devices can communicate with each other, the quorum sets is so chosen that any two devices will wake up and connect with each other at certain time slots. In other words, there are always nonempty intersections between the quorum sets of individual devices.

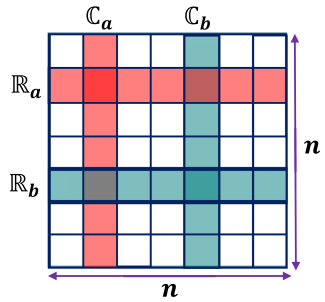


Fig. 9. A grid-based quorum.

Definition 6.2 (Grid-based Quorum). In a grid-based quorum, each continuous n^2 time slots is considered as a quorum set, which are arranged in a $n \times n$ grid. A square box in a grid represents a time slot. Each device independently chooses one row and one column from the $n \times n$ grid as a quorum set.

The concept of grid-based quorum is illustrated in Figure 9, in which device A picked row $\mathbb{R}_a$ and column $\mathbb{C}_a$ as its quorum, while device B picked row $\mathbb{R}_b$ and column $\mathbb{C}_b$. This figure also shows that there are two intersections between the quorum sets of devices A and B—one for $\mathbb{R}_a$ and $\mathbb{C}_b$ and the other for $\mathbb{C}_a$ and $\mathbb{R}_b$. As mentioned, the devices must wake up at the chosen quorums, which means both of these devices will wake up at these intersections and communicate. *However such quorum-based grid allocation cannot be directly applied to WiFi-tethering-based communication*, due to the additional requirement that the devices need to wake up in different modes. This gives rise to the need for developing a new grid-based quorum set generation scheme. The aim is to ensure that a device can *discover* its neighboring devices in spite of the mobility as long as there is a valid overlap between any devices within a bounded amount of time.

Definition 6.3 (Valid Overlap). A *valid overlap* in a time slot happens between two devices when one of them is in the hotspot mode and the other in the client mode.

To ensure that the devices share some valid overlapping slots, we need to enforce three principles for the quorum grid formulation. Our problem is different from the neighbor discovery schemes in sensor networks [45–47] in the following sense. Unlike the related works where an overlap ensures a successful discovery, our scheme needs to ensure a *valid* overlap between two devices implies a successful discovery. Below, we describe three grid allocation principles for successful WiFi-tethering-based communication.

(A) First rule: Each row of a grid consists of n consecutive (mod n^2) slots. For example, in Figure 10, the rows consist of $n = 4$ consecutive slots, i.e., (1. . .4), (5. . .8), (9. . .12), (13. . .16).

(B) Second rule: $\forall m \in [1, 2, \ldots, n^2]$, any n continuous slots

$$[m, m+1, \ldots, m+n-1] \pmod{n^2} \tag{4}$$

are distributed in n different columns. For example, in Figure 10, with $m = 2$, the slots (2. . .5) are distributed to n columns.

Using these two simple grid formation principles any neighboring devices have some intersecting slots, which is proven in Reference [45].

(C) Third rule: For WiFi-tethering-based communication, it is necessary that two devices are in different modes (i.e., hotspot and client) in their intersecting slots. For this purpose, each of the devices has two disjoint quorum sets: one for hotspot mode and another for client mode. Two disjoint quorum sets can be easily picked up from the $n \times n$ grid such that the corresponding rows

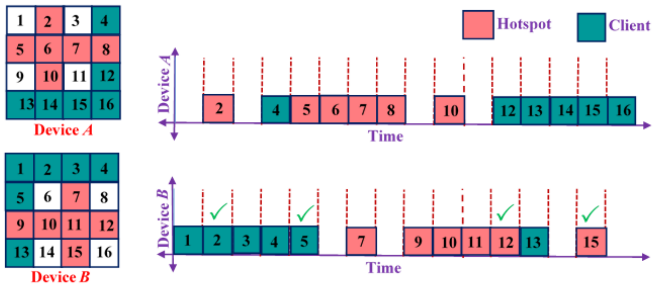


Fig. 10. An example of intersecting hotspot and client slots for devices A and B.

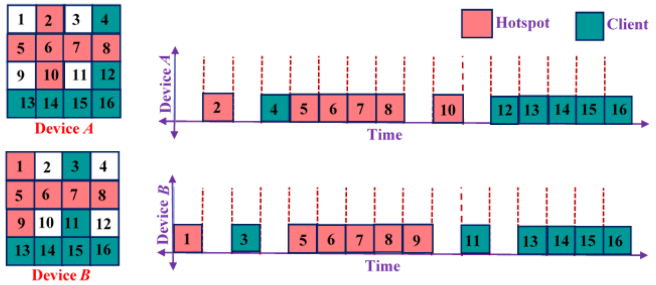


Fig. 11. An example when devices A and B do not overlap within a cycle.

and columns are different. There will be two slots common to both hotspot and client modes, which we call the *common slots (CSs)*. In such scenarios, *the state representing the row will win*. In Theorems 6.5–6.8, we show that by using these principles any neighboring devices always experience some valid overlaps.

For example, in Figure 10, device A chooses (row2, column2) and (row4, column4) as hotspot and client sequence, resulting in two common slots 8 and 14 chosen as hotspot and client, respectively.

Definition 6.4 (Cycle). The *cycle* of a $n \times n$ grid is n^2, which is 16 for 4×4 grid in Figure 10. Thus, after 16 slots, the device A repeats the same sequence.

6.1 Probability of No Valid Quorum

Using the above grid formation principles, two devices *generally* find a valid overlap within a bounded amount of time. For example, in Figure 10, nodes A and B intersect in hotspot and client mode (and vice versa) in slots 2, 5, 12, and 15. However, there lies a special case where two devices do not find a valid overlap. Such a scenario happens if the following three conditions are satisfied: (a) Two devices A and B have identical grid dimensions; (b) They choose exactly the same rows for their hotspot and client states; and (c) A's hotspot column does not overlap with B's client column and vice versa.

In such a scenario, the client slot of a device never overlaps with the hotspot slot of the other device and vice versa. This case is depicted in Figure 11, where both A and B's grid dimensions are 4×4, satisfying condition (a) above. Also device A chooses (row2, column2) and (row4, column4) as hotspot and client sequence, whereas B chooses (row2, column1) and (row4, column3) as hotspot and client sequence, respectively, thus conditions (b) and (c) are also satisfied. However, such quorum set generation does not generate a valid overlap between A and B. Below, we prove that this special scenario occurs with very low probability.

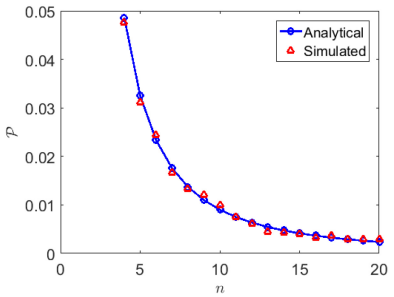


Fig. 12. Probability of not overlapping with n.

Theorem 6.5. *In a homogeneous (or identical) grid dimension $n \times n$, two devices do not experience a valid overlap with a probability of $\mathcal{P} = \mathcal{P}_1\mathcal{P}_2$, where $\mathcal{P}_1 = \frac{1}{n(n-1)}$ and $\mathcal{P}_2 = \frac{n^2-3n+3}{n^2(n-1)}$.*

Proof. The proof is in Appendix A.1. □

We now claim that $\mathcal{P}$ is extremely low even for small n. Figure 12 shows the variation of $\mathcal{P}$ with different n, which validates our analytical expressions against the simulation results. From this figure, we can observe that even with $n = 4$, $\mathcal{P}$ is less then 5%, and even lower for higher n. We thus claim that using the above three grid formation principles, the devices experience valid overlaps within a bounded time *almost surely*. As the number of mobile users in a geographic region is assumed to be high, the devices are expected to remain connected even if a few device-pairs do not experience valid overlaps.

Theorem 6.6. *Assume that two devices A and B use the identical grid size of $n \times n$. Following the grid allocation rules, A's hotspot schedule overlaps with B's client schedule at least twice within every n^2 consecutive quorum slots, if the special case proposed in Theorem 6.5 does not arise.*

Proof. The proof is in Appendix A.2. □

Lemma 6.7. *Any device goes into the client (or hotspot) mode at least once in $2n$ consecutive slots.*

Proof. In a client column, there are two *potential* client slots in $2n$ consecutive slots. One of them may be a hotspot slot due to common slot while the other one has to be a client slot. Similar logic applies for the hotspot slots as well. □

Theorem 6.8. *Using the grid formation rules, if the special case of Theorem 6.5 does not arise, then any pair of devices will overlap in a hotspot-client or client-hotspot mode at least twice in $2.max\{m^2, n^2\}$ slots, where m and n are the grid dimensions of the two devices.*

Proof. The proof is in Appendix A.3. □

Notice that for a heterogeneous grid dimension, any two devices have at least two *guaranteed* valid overlaps within $2.\Upsilon$ slots as shown in Theorem 6.8. Only in case of a homogeneous grid dimension, there is a *small* probability that two devices do not experience a valid overlap, as claimed in Theorem 6.5.

The proofs of Theorem 6.6 and Theorem 6.8 do not assume any node synchronization. Thus, the neighboring devices still experience valid overlaps within the bounded time almost surely, even in the absence of node synchronization.

6.2 Determining Grid Dimensions

In general, the grid sizes for different devices can be chosen as either the same or differently. The problem with the identical (or homogeneous) grid dimension is that it does not distinguish between the remaining battery capacity of different devices. We introduce this distinction by choosing the device sleep periods based on their remaining battery capacity. According to the quorum-grid formation, a device using $n \times n$ grid stays awake for $4n - 4$ out of n^2 time slots (2 rows + 2 columns - 4 common slots), which we define as *active slots*, and sleeps for the remaining slots. Thus, the desired behavior can be achieved by choosing a larger grid size for devices with smaller remaining battery capacity, and vice versa.

For example, we can categorize the devices into four types, such as High (H), Medium (M), Low (L), and Very low (VL), based on their battery powers. Assume that the devices are considered to be of high power if their batteries are >75% charged. Similarly the devices of other classes are in between (50–75%), (25–50%), and (0–25%) charged, respectively. Thus, based on these thresholds (100:75:50:25), the ratios of their active slots need to be 4:3:2:1. If the grid size for H is chosen to be $h \rightarrow 5{\times}5$, then m, l, v corresponding to M, L, VL types are chosen as follows:

$$4h - 4 \;/\; h^2 : 4m - 4 \;/\; m^2 = 4 : 3 \Rightarrow m \rightarrow 7 \times 7, \tag{5}$$

$$4h - 4h^2 : 4l - 4 \;/\; l^2 = 4 : 2 \Rightarrow m \rightarrow 10 \times 10, \tag{6}$$

$$4h - 4 \;/\; h^2 : 4v - 4 \;/\; v^2 = 4 : 1 \Rightarrow m \rightarrow 20 \times 20, \tag{7}$$

which ensures 64%, 49%, 36%, and 9% active times, respectively. By choosing the grid size based on each device's individual battery charge, the E-DARWIN framework ensures that the smartphones with low battery can proportionately decrease their battery drain rate and thus can continue to participate in the network.

7 GRID QUORUM-BASED DATA FORWARDING

A device transmit frequent beacon messages in hotspot mode to allow its neighboring devices to associate with it. The device maintains a *neighbor table* storing its neighbors, path-duty-cycles (path-DCs) (defined below), and time-stamp when it was last associated with those neighbors. If no association happens with a neighbor for a long time (defined as *discard interval (DI)*), then its entry from the table is deleted. The smartphones also update their neighbor table and path-DCs whenever a new association takes place. *The devices also exchange their wake-up schedules (hotspot and client schedules) and local clocks with their neighbors during the association.* The wake-up schedules of the neighbors are utilized in the forwarding phase as mentioned later. The local clock of a device is used by its neighbors to find out its active slots relative to their own. The devices also share whether they are mostly mobile or static in their beacon messages. The beacon messages typically carry the following fields: ⟨nodeID, path-DC, hotspot/client schedule, local-clock, mobile/static⟩.

Definition 7.1 (Path-duty-cycle). The *path-duty-cycle* (or *path-DC*) is the product of the DCs of all the devices along a path. A device with high duty-cycle has high energy availability and goes into hotspot and client modes more frequently. Thus, the path-DC reflects how good a route is in terms of the entire path latency. Higher path-DC means the intermediate devices have higher battery capacity (as the duty-cycle is proportional to battery-capacity) and vice versa.

We propose two types of forwarding mechanisms: energy-delay aware and hot-potato-based, depending on whether a device is connected to a network, or disconnected but trying to reconnect.

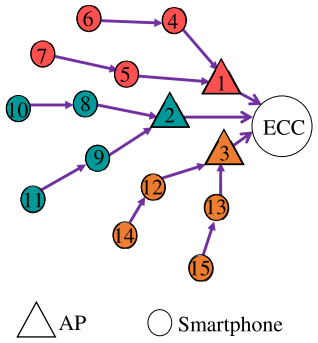


Fig. 13. Illustrating smartphone-AP association.

7.1 Energy-delay Aware Forwarding

The path selection is performed based on maximizing the path-DC, i.e., minimizing the overall packet latency. Note that the longer the path, the lesser is the path-DC, i.e., the metric prefers shorter paths. However, the routes always prefer the devices with higher energy availability or shorter n in this process, where n is the number slots.

This is achieved as follows. The APs always broadcast a path-DC = 1. Each device calculates its path-DC as that of its parent multiplied by its own DC, and then broadcasts. A device i chooses j as its parent among all its neighbors if

$$\text{path-DC of } j > \text{path-DC of } k \quad \forall k \neq j. \tag{8}$$

In this process, a device chooses a parent with the maximum path-DC value (ties are broken randomly) such that the beacon signal strength from that device is higher than some threshold Γ. Thus, the devices effectively form a tree like structure towards their APs as shown in Figure 13. Repetitive collisions are avoided by using the binary exponential backoff mechanism used in the traditional 802.11 standard.

Recall that the wake-up schedules proposed in Section 6 ensure that the devices can discover one another within a bounded time. However, just transmitting packets in those slots will result in significantly large packet delays, especially in low duty-cycle scenarios, which is unexpected in a disaster scenario. For example, if two devices with homogeneous grid dimensions overlap in four out of n^2 slots, for $n = 10$ and a slot duration of 1 minute, then the devices may experience only two valid overlaps (applying Theorem 2) in 100 minutes (or one overlap in ~50 minutes). For a multi-hop network, this will result in significantly high latency. However, reducing the grid dimensions (or increasing the duty-cycles) is not energy-efficient as there are just two valid overlaps out of $4n - 4$ active slots. We also believe that the slot duration should not be chosen very small (say, less than 1 minute), because in WiFi-tethering the state transitions take ~1–5 seconds, which is much higher than the radio on-off time (typically in milliseconds) in low-power wireless sensor networking applications.

We thus adopt a forwarding policy where the devices record the active slots of their neighbors (when they meet one another in their valid overlapped slots), and turn on their radios in their parent's active slots (in opposite mode) to exchange the data packets. If a device cannot reach its parent in a particular slot, then it retries in the successive slots. If a retransmission limit Ψ is reached (i.e., the parent or the device may go out of their range), then it decides on its next best neighbor as its parent and sends its data packets in the corresponding slots. Figure 14 shows the average and maximum delay a device needs to wait for an active slot of any neighbor (or parent) with different grid dimensions (assuming they use identical grid dimensions). From this

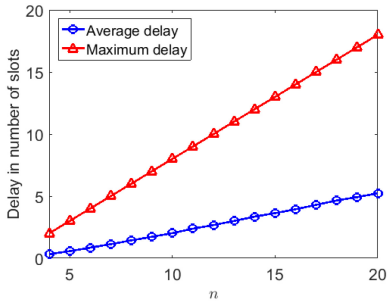


Fig. 14. Delay vs. number of slots (n).

figure, we can observe that with $n = 10$ (i.e., ∼40% duty cycle), the average waiting time is ∼2 slots. Such a forwarding policy significantly reduces the end-to-end packet latency and makes the scheme applicable to practical environment for collecting crucial information about situational awareness, and at the same time let the devices sleep most of the time. Notice that *the energy-delay aware forwarding mechanism is used only by the devices that are mostly static*; the mobile devices use an opportunistic forwarding strategy discussed next.

7.2 Hot-potato-based Forwarding

In case of heterogeneous gird formations, different devices choose their grid sizes differently (see Section 6.2). Let us assume that the largest grid dimension allowed for any device is predefined and assumed to be $n_{\max} \times n_{\max}$.

Let us now describe the hot-potato-based forwarding scheme in which a mobile device sends its stored packets to one of its static neighbors. To do so, it chooses a grid dimension of $n_{\max} \times n_{\max}$, and remains on for $2n_{\max}$ consecutive time slots in the client or hotspot mode. If it can connect to any neighbor (for multiple hotspot neighbors, one of them will be selected randomly), then it uploads its packets. Note that using the grid formation rules, any device goes into its hotspot (or client) mode at least once in $2n_{\max}$ slots (using lemma 6.7). Thus, in $2n_{\max}$ slots the device can connect to all its neighboring devices. A device can also quickly discover and learn the schedules of its neighbors by switching to this state. This is required when a device is frequently moving and can still send its packets successfully to the AP. We strategically choose $n_{\max} = 10$ (we isolate the VL phones from forming a multi-hop tree as they will die fast). Thus, the mobile devices choose any 20 out of 100 slots to remain active in one of the two modes (hotspot or client). In this process, the mobile devices keep their duty cycle low, and still can send their packets to one of their static neighbors (if any) with bounded delay.

7.3 Efficient Handing of Device Mobility

In a realistic disaster environment, people frequently move depending on their needs; hence, the application should consider some practical adjustments because of the joining or departure of devices from their corresponding network clusters. When a device is connected and does not move frequently, it uses energy-delay aware forwarding to send its packets to its parent. Whenever the device does not find any neighbor for more than some time, T, it starts a rediscovery process where it switches to *searching state*. In this state, the device adopts hot-potato-based forwarding strategy and neighbor discovery. If it finds any neighbor in this process, then it joins the network; otherwise, it backs-off and starts the rediscovery after certain interval, as shown in Figure 15. If a device joins and leaves the network very frequently, then it can be inferred that the person is in a highly *mobile* state, when the device uses hot-potato-based forwarding and discovery process

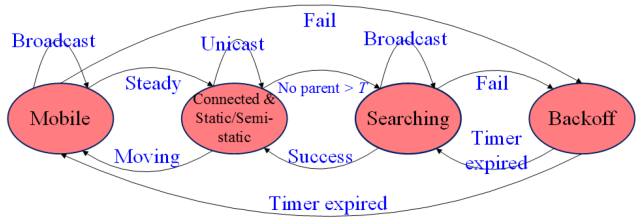


Fig. 15. State transition diagram.

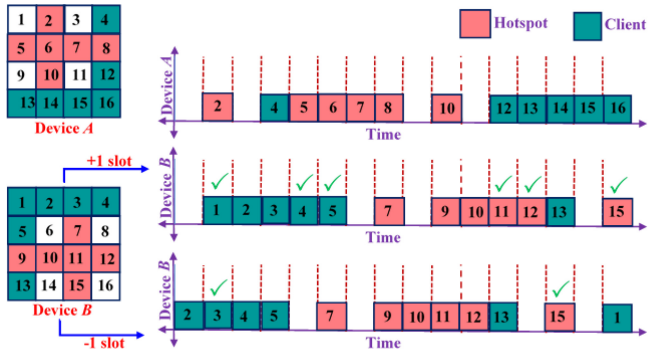


Fig. 16. An illustration of clock desynchronization.

to join the network. If it fails to find any neighbor in this process, then the application stops the network operation (since frequent joining/leaving is expensive for the device's battery), goes into the dormant state and reinitiates the discovery process after certain backoff.

Finally, we address the issue of time synchronization across smartphones for the scheme to work properly. The main impact of the relative clock skews in smartphones is that a device may wake-up earlier or later than the schedule of its parents and thus their slots fail to overlap enough for the communication to take place. However, given the slot duration in minutes, very crude synchronization is adequate. For example, it is shown in Reference [48] that synchronizing the devices once every 12 hours is enough to bound the clock drift to 100 ms in the worst case. Thus, assuming that the devices were synchronized before the disaster struck (e.g., using the Network Time Protocol (NTP)) and the disconnection does not last more than a few days, time drift is not an issue. For longer outages, it is possible to exploit Global Positioning System (GPS)-based time synchronization if needed. Figure 16 shows the effect of clock desynchronization where the clock of device B is leading/trailing by one slot. Still the devices experience at least two valid overlaps in a cycle as stated in Theorem 2.

8 PERFORMANCE EVALUATION

We evaluate the performance of E-DARWIN through prototyping and extensive simulations.

8.1 Power Consumption Evaluation

We implemented a prototype of E-DARWIN on a Samsung Galaxy S III device running Android OS 4.4.2 to measure the power consumption of the device (using Power Monitor [49]) during discovery. The results were used as guidelines for selecting various algorithm parameters, which are then used in simulation experiments. The results reported in the following are average over 25 measurements for each experiment.

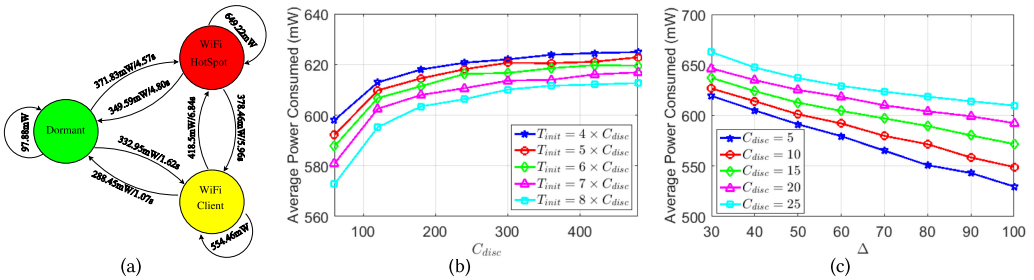


Fig. 17. (a) Power and Time Consumed in State Transitions. Average power consumed in the initialization phase with varying (b) T_{init} and C_{disc}, (c) C_{disc} and Δ.

8.1.1 Power Consumption of Different States. In the first set of experiments, we measured power consumption and residence time in each state. As shown in Figure 17(a), WiFi hotspot is the most power-intensive state. In the hotspot mode, the devices consumes ~649.22 mW power, which is ~6–7 times that of the dormant state. More energy and time is also consumed when transitioning to or from the WiFi hotspot state. This is because a transition into or out of WiFi hotspot mode requires enabling/disabling of wireless interface, retrieval/storage of the hotspot configuration, and creation/destruction of a wireless network with the given configuration. These operations require significant amount of I/O operations and processing, which result in higher energy consumption. When in the hotspot state, the device always stays in the active state as it repeatedly broadcasts network discovery beacons, and performs various network operations as well as maintenance, making it the most power exhaustive state.

8.1.2 Effects of T_{init}, C_{disc} and Δ. We analyze the impact of various algorithmic parameters on the device power consumption in the initialization phase. First, we increase the initialization time (T_{init}) while keeping $C_{\text{disc}} = 10$ and $\Delta = 30s$. As shown in Figure 17(b), the devices consume more power with increasing $\frac{T_{\text{init}}}{C_{\text{disc}}}$. This is because, with increasing $\frac{T_{\text{init}}}{C_{\text{disc}}}$, the devices transition in between the hotspot and client modes more frequently, thereby increasing the power consumption. While a high ratio of $\frac{T_{\text{init}}}{C_{\text{disc}}}$ increases the probability of a device discovering its neighbor (as $\rho \propto \frac{T_{\text{init}}}{C_{\text{disc}}}$), it also results in higher power consumption for the devices. Conversely, increasing C_{disc} with constant T_{init} (assumed to be 10^4 seconds) reduces the power consumption as the devices transit in between hotspot and client state less frequently, as shown in Figure 17(c). Similarly, the power consumption decreases with increasing Δ. But this will also reduce the probability of two devices discovering one another as there is a higher probability that they repeatedly become clients at the same time.

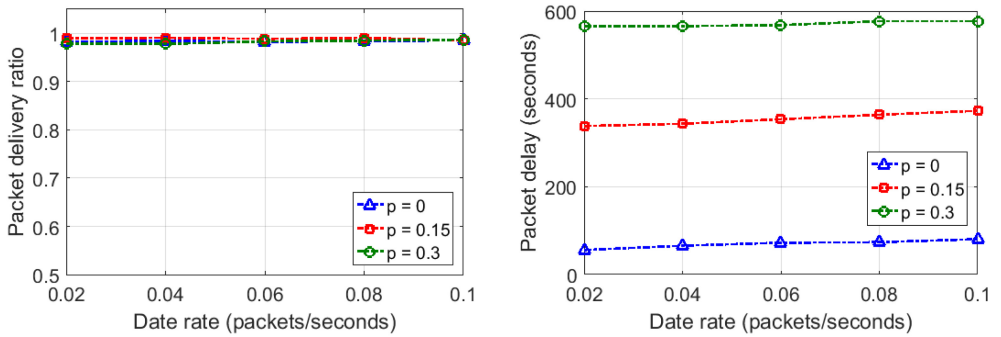
8.2 Simulation-based Network Performance Evaluation

8.2.1 Simulation Model Calibration. We studied the performance of the proposed quorum-based data forwarding scheme in *Castalia* (http://castalia.npc.nicta.com.au), which is an application-level simulator for wireless sensor networks based on OMNeT++. The simulated system topology consists of 100 devices including 5 APs, placed uniformly in an area of 200×200 sq. m. In the presence of device mobility, this fixed topology changes according to the movement pattern of the devices.

We assume a log-normal shadowing model with path-loss exponent $k = 2.4$, and standard deviation $\sigma = 4$ dBm. The path loss at a reference distance $d_0 = 1$ is assumed to be of 55 dBm. We assume additive interference model for our simulations. The transmit power is assumed to be of 20 dBm, which is typical in current smartphones. The signal-delivery threshold and Clear Channel Assessment (CCA) threshold are assumed to be of −80 dBm and −95 dBm, respectively. We

Table 2. Simulation Parameters

Radio Propagation Model	Log-normal Shadowing Model	**Transmission Power**	20 dBm
Path Loss Exponent	2.4	**Shadowing standard deviation**	4 dBm
Signal delivery threshold	−80 dBm	**CCA threshold**	−95 dBm
MAC Protocol	IEEE 802.11	**Antenna Model**	Omni-directional



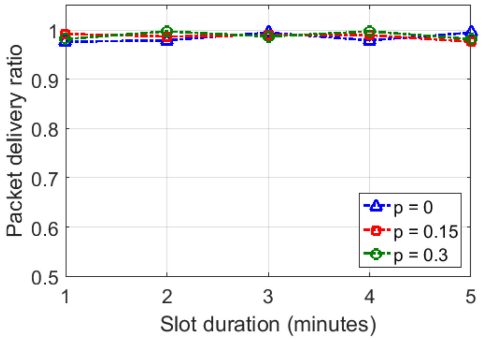
(a) Packet Delivery Ratio vs. Transmission Rate (b) Packet Delay vs. Transmission Rate

Fig. 18. Packet Delivery (a) and Packet Delay (b) vs. Transmission Rate.

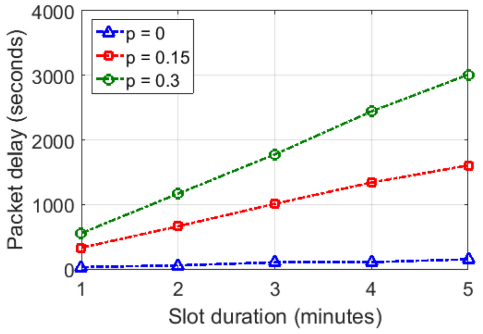
assume $\Gamma = -73$ dBm to ensure high possibility of successful packet reception at the receiver end. Key parameters used for the simulations are listed in Table 2.

We evaluate our data forwarding scheme by measuring the key network performance metrics, such as packet delivery ratio and cumulative network delay incurred by a packet to reach from a device to an AP. We consider scenarios with static devices as well as mobile devices that move in a random direction at a certain speed (nominally 1 m/s) with probability of p. At the boundaries of the geographic field, the nodes bounce off in a different random direction. We assume that the devices adopt carrier sense multiple access / collision avoidance (CSMA/CA) as their access protocol, with a maximum retransmission count of 30. We deactivate the RTS/CTS (request to send / clear to send) mechanism to conserve the device's remaining energy. The remaining battery capacities of the devices are uniformly randomly chosen between (25%–100%) of a fully charged device. Unless otherwise mentioned, we choose quorum grid-sizes of 5×5, 7×7, and 10×10 for devices with (75–100%), (50–75%), and (25–50%) battery capacities, respectively. This means that the devices spend 64%, 49%, and 36% times in their active modes, respectively. For all our experiments, we assume a low data rate in between 0.01–0.1 packets/second (one packet in 10–100 seconds), which is sufficient for our application.

8.2.2 Effect of Packet Transmission Rates. Figure 18 shows the packet delivery ratio and packet delivery latency with the variation of packet transmission rates, which are assumed to be constant for all devices. The slot duration is assumed to be 1 minute. We can observe a slight increase in the average packet delay while the data rate increases from 0.02 packets/second to 0.1 packets/second. However, the packet delay increases by ∼7–10 times, with the increase in device mobility. This is because the mobile devices follow the hot-potato-based forwarding strategy, while they remain awake infrequently to transmit their stored packets. This increases the time for valid overlaps with their static neighbors, which drastically increases the packet delivery delay. We can also observe that the packet delivery ratio does not vary even in the presence of device mobility, which shows the effectiveness of our proposed forwarding scheme.

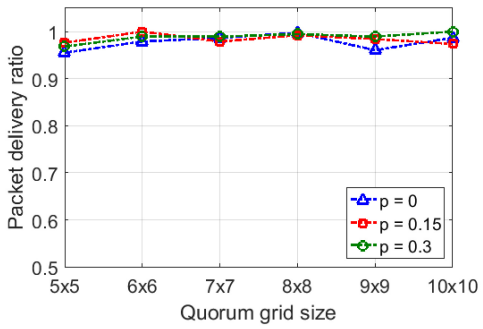


(a) Packet Delivery Ratio vs. Slot Sizes

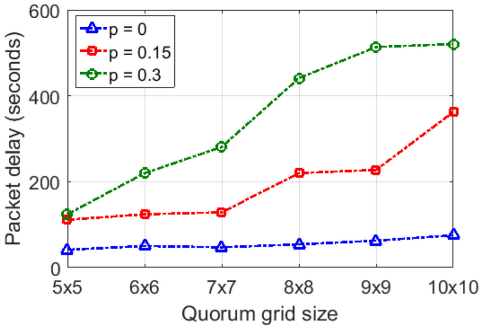


(b) Packet Delay vs. Slot Size

Fig. 19. Packet Delivery (a) and Packet Delay (b) vs. Slot Size.



(a) Packet Delivery Ratio vs. Grid Sizes



(b) Packet Delay vs. Grid Size

Fig. 20. Packet Delivery (a) and Packet Delay (b) vs. Grid Size.

8.2.3 Effect of Quorum Slot Durations. Figure 19 shows the delivery ratio and packet delay as a function of slot durations. The packet transmission rate is assumed to be 0.01 packets/second. We can observe that the packet delay increases ~3–6 times and almost linearly with the increase in slot duration. This is due to the increasing delay in between the valid overlaps of the devices. The packet delay also increases by ~10–20 times with increased mobility. The packet delivery ratio also remains constant and above 98% in all such experiments. By keeping the slot duration small (~1 minute), the packet delivery latency can be limited to ~10 minutes, which is sufficient for our data collection application.

8.2.4 Effect of Grid Lengths. Figure 20 shows the delivery ratio and packet delivery delay for various quorum grid sizes, assuming ther are identical for the devices. The packet transmission rate is assumed to be 0.01 packets/second, whereas the slot duration is kept as 1 minute. Observe that the packet delay increases by ~3–5 times as the grid size increases from 5×5 to 10×10, especially in the presence of device mobility. This is because a device's duty-cycle decreases with increase in grid dimensions, thus resulting in higher waiting time for forwarding the stored packets to the parents. In Figures 18–20, the average packet delay is in hundreds of seconds, which is sufficient for building a basic infrastructure with the devices in a disaster-hit area to the ECC. Once such infrastructure is established, we can use efficient mechanisms like virtual circuits for bulk data transfers [50], the details of which are beyond the scope of this article.

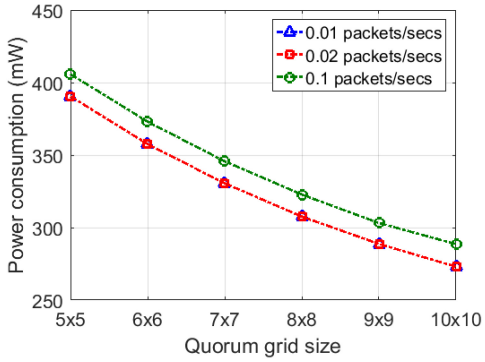


Fig. 21. Power consumption of E-DARWIN devices in data forwarding stage.

8.2.5 Power Consumption in Data Forwarding Phase. Figure 21 shows the average power consumption of the E-DARWIN devices in the data forwarding phase for different quorum grid sizes. This figure assumes that the grid sizes are identical, whereas the slot duration is 1 minute, and that the devices are static. From Figure 21, we can observe that the power consumption decreases by ∼30% when the grid size increases from 5×5 to 10×10. This is because the devices stay awake for $4n - 4$ out of n^2 slots, and stay dormant in the remaining slots. Moreover, the power consumption increases marginally ∼5% when the transmission rate increases by 10 times. This is mainly due to the extra power consumption during state transitions, as also observed from Figure 17(a).

9 CONCLUSIONS

In this article, we presented an architecture, called E-DARWIN for creating an ad hoc network infrastructure in disaster affected regions using WiFi tethering. The proposed solutions were analyzed through a comprehensive set of experiments using a prototype implemented on Android platform and large-scale simulations. Our future plan is to consider mobile devices with a wider set of characteristics in terms of available sensors, networking capabilities, and storage availability. We will also examine the security issues and tradeoff between security and performance.

A APPENDIX

A.1 Proof of Theorem 6.5 (see page 15)

Proof. For homogeneous (or identical) grid dimension, two devices A and B do not experience a valid overlap if (a) they choose exactly the same rows for their hotspot and client states, which occurs with probability of $\mathcal{P}_1$, and (b) A's hotspot column does not overlap with B's client column or vice versa, which happens with probability $\mathcal{P}_2$.

Let us now derive the expression for $\mathcal{P}_1$. Note that out of n rows, two devices choose the same rows as hotspot state with probability $\binom{n}{1}.(\frac{1}{n})^2 = \frac{1}{n}$. From the remaining rows, they choose the same ones as client states with probability $\binom{n-1}{1}.(\frac{1}{n-1})^2 = \frac{1}{n-1}$. Hence,

$$\mathcal{P}_1 = \frac{1}{n(n-1)}. \tag{9}$$

Next, we derive the expression for $\mathcal{P}_2$. For simplicity, let us assume that A and B choose their column slots in the following sequence: (1) A first chooses its hotspot column, then (2) B chooses its client column, next (3) A chooses its client column, and finally (4) B chooses its hotspot column.

First A chooses its hotspot column from any of the grid columns. After that B chooses its client column randomly, which does not overlap with A's hotspot column with probability $\frac{n-1}{n}$. Now A chooses its client column from the remaining $n-1$ columns (except its hotspot column). There are two cases to consider:

(i) A and B's client columns are identical, which occurs with probability $\frac{1}{n(n-1)}$,
(ii) A and B's client columns are different, which occurs with probability $\frac{n-2}{n(n-1)}$.

In case (i), irrespective of whatever hotspot column B chooses, it does not overlap with A's client column. In case (ii), B's hotspot column does not overlap with A's client column with probability of $\frac{n-2}{n-1}$. Putting everything together,

$$\mathcal{P}_2 = \frac{n-1}{n}\left[\frac{1}{n(n-1)} + \frac{n-2}{n(n-1)}.\frac{n-2}{n-1}\right] = \frac{n^2-3n+3}{n^2(n-1)}. \quad \square \tag{10}$$

A.2 Proof of Theorem 6.6 (see page 15)

Proof. The special case mentioned in Theorem 6.5 arises if (a) the devices A and B choose exactly the same rows for their hotspot and client states, and (b) A's hotspot column does not overlap with B's client column or vice versa. First assume that condition (a) is not satisfied as A and B choose different rows ($\mathbb{R}_a$ and $\mathbb{R}_b$, respectively, $\mathbb{R}_a \neq \mathbb{R}_b$) for their hotspot states. In this case, the client column of B (i.e., $\mathbb{C}_b$) will definitely experience a valid overlap with $\mathbb{R}_a$ as *the overlapping slot is not going to be the common slot for B*. A similar overlap occurs in between the hotspot row of B and the client column of A. Thus, there will be at least two overlaps within one cycle.

However, if condition (b) is not satisfied, then A's hotspot column overlaps with B's client column or vice versa. In this case, there will be at least $n-2$ valid overlaps (disregarding two possible common slots of two devices). Thus, the theorem is proved. $\square$

A.3 Proof of Theorem 6.8 (see page 15)

Proof. First consider the case of homogeneous grid, i.e., $m = n$. In this case, by applying Theorem 6.5, we can show that there will be at least two overlapping hotspot-client combinations within m^2 slots. Next, consider the case of heterogeneous grid dimensions m and n of devices A and B, respectively. Without loss of generality, assume $m > n$. We further assume that A chooses row $\mathbb{R}_a$ and column $\mathbb{C}_a$ for its hotspot mode, while B picks row $\mathbb{R}_b$ and column $\mathbb{C}_b$ for its client mode.

Now consider the hotspot slots of $\mathbb{R}_a$ and client slots of $\mathbb{C}_b$. $\mathbb{R}_a$ consist of m consecutive hotspot slots (due to second and third rules). However, in case of $\mathbb{C}_b$, except one slot (assume this as s_1, which is changed to hotspot as it is a common slot), others repeat with an equal difference of n. Since $m > n$, there is at least one overlap between $\mathbb{R}_a$ and $\mathbb{C}_b$. However, as s_1 is changed to the hotspot mode, there may not be a situation where there is no *valid* overlap between A and B within $\Upsilon = \max\{m^2, n^2\}$ slots, as shown in Figure 22.

In such a situation, there can be two case. In the first case, m^2 is not a multiple of n^2, implying the starting point t of A's next cycle cannot be the starting point of B's cycle. According to the grid formation principle, there will be at least one overlap between $\mathbb{R}_a$ and $\mathbb{C}_b$ in this cycle. Assume that this slot is s_2. However, as t is not the starting point of B's cycle, $s_2 \neq s_1$ and thus s_2 cannot be the common slot of B. Therefore, there will be a valid overlap at s_2 and at least one valid overlap within $2.\Upsilon$ slots.

Let us now consider the second case where m^2 is a multiple of n^2. In this scenario, m consecutive hotspot slots of A will overlap with at least one client slot of B other than s_1. Thus, there will be

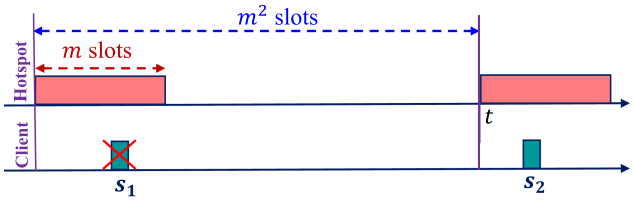


Fig. 22. An illustrative example for Theorem 6.8.

at least one valid overlap within Υ slots. Similarly, there will be at least one valid overlap within $2.\Upsilon$ slots because of the client rows and hotspot columns of A and B, respectively. □

ACKNOWLEDGMENTS

The authors are grateful to the anonymous referees for insightful comments and constructive suggestions, which helped us improve the quality of the manuscript significantly.

REFERENCES

[1] Yilang Wu, Amitangshu Pal, Junbo Wang, and Krishna Kant. 2019. Incremental spatial clustering for spatial big crowd data in evolving disaster scenario. In *Proceedings of the IEEE CCNC*. 1–8.

[2] Shanshan Zhang, Amitangshu Pal, Krishna Kant, and Slobodan Vucetic. 2018. Enhancing disaster situational awareness via automated summary dissemination of social media content. In *Proceedings of the IEEE GLOBECOM*. 1–7.

[3] Yilang Wu, Krishna Kant, Shanshan Zhang, Amitangshu Pal, and Junbo Wang. 2017. Disaster network evolution using dynamic clustering of twitter data. In *Proceedings of the IEEE ICDCS Workshops*. 348–353.

[4] Amitangshu Pal, Junbo Wang, Yilang Wu, and Krishna Kant. Big Data Analysis in Disaster Management Networks: A Short Tutorial. [n.d.]. In submission.

[5] A. F. Aji, I. P. E. S. Putra, P. Mursanto, and S. Yazid. 2014. Can smartphones be used to detect an earthquake? Using a machine learning approach to identify an earthquake event. In *Proceedings of the SysCon*. 72–77. DOI: http://dx.doi.org/10.1109/SysCon.2014.6819238

[6] J. Reilly, S. Dashti, M. Ervasti, J. D. Bray, S. D. Glaser, and A. M. Bayen. 2013. Mobile phones as seismologic sensors: Automating data extraction for the ishake system. *IEEE Trans. Autom. Sci. Eng.* 10, 2 (April 2013), 242–251. DOI: http://dx.doi.org/10.1109/TASE.2013.2245121

[7] A. Zambrano, I. Perez, C. Palau, and M. Esteve. 2014. Quake detection system using smartphone-based wireless sensor network for early warning. In *Proceedings of the PERCOM Workshops*. 297–302. DOI: http://dx.doi.org/10.1109/PerComW.2014.6815221

[8] Akira Nishimura. 2014. Encoding data by frequency modulation of a high-low siren emitted by an emergency vehicle. In *Proceedings of International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. 255–259. DOI: http://dx.doi.org/10.1109/IIH-MSP.2014.70

[9] Koichiro Takeuchi, Tetsuya Matsumoto, Yoshinori Takeuchi, Hiroaki Kudo, and Noboru Ohnishi. 2014. A smart-phone based system to detect warning sound for hearing impaired people. In *Computers Helping People with Special Needs*. Lecture Notes in Computer Science, Vol. 8548. Springer International Publishing, 506–511.

[10] Konrad Lorincz, David J. Malan, Thaddeus R. F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. 2004. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervas. Comput.* 3, 4 (2004), 16–23.

[11] Stephen M. George, Wei Zhou, Harshavardhan Chenji, MyoungGyu Won, Yong Oh Lee, Andria Pazarloglou, Radu Stoleru, and Prabir Barooah. 2010. DistressNet: A wireless ad hoc and sensor network architecture for situation management in disaster response. *IEEE Commun. Mag.* 48, 3 (2010), 128–136.

[12] Mohamed Younis, Sookyoung Lee, and Ameer A. Abbasi. 2010. A localized algorithm for restoring internode connectivity in networks of moveable sensors. *IEEE Trans. Comput.* 59, 12 (Dec. 2010), 1669–1682. DOI: http://dx.doi.org/10.1109/TC.2010.174

[13] Jianyong Lin, Wendong Xiao, F. L. Lewis, and Lihua Xie. 2009. Energy-efficient distributed adaptive multisensor scheduling for target tracking in wireless sensor networks. *IEEE Trans. Instrument. Measure.* 58, 6 (June 2009), 1886–1896. DOI: http://dx.doi.org/10.1109/TIM.2008.2005822

[14] Weifeng Shan, Jilin Feng, Jianjun Chang, Fan Yang, and Zhonghua Li. 2012. Collecting earthquake disaster area information using smart phone. In *Proceedings of the ICSSE*. 310–314.

[15] Peter W. Gething and Andrew J. Tatem. 2011. Can mobile phone data improve emergency response to natural disasters? *PLoS Med.* 8 (2011).
[16] Weiquan Lu, Winston K. G. Seah, Edwin W. C. Peh, and Yu Ge. 2007. Communications support for disaster recovery operations using hybrid mobile ad-hoc networks. In *Proceedings of the LCN*. 763–770.
[17] Niranjan Kumar Ray and Ashok Kumar Turuk. 2011. A framework for disaster management using wireless ad hoc networks. In *Proceedings of the CCS*. 138–141.
[18] T. Al Hadhrami, Qi Wang, and C. Grecos. 2013. Power- and node-type-aware routing algorithm for emergency-response wireless mesh networks. In *Proceedings of the VTC*. 1–5. DOI: http://dx.doi.org/10.1109/VTCSpring.2013.6692718
[19] Shivashankar, H. N. Suresh, G. Varaprasad, and G. Jayanthi. 2014. Designing energy routing protocol with power consumption optimization in MANET. *IEEE Trans. Emerg. Top. Comput.* 2, 2 (June 2014), 192–197. DOI: http://dx.doi.org/10.1109/TETC.2013.2287177
[20] B.A.T.M.A.N.—Better Approach to Mobile Ad-hoc Networking. [n.d.]. Retrieved from http://www.open-mesh.org/projects/open-mesh/wiki.
[21] Josh Thomas, Jeff Robble, and Nick Modly. 2012. Off grid communications with android meshing the mobile world. In *Proceedings of the IEEE HST*. 401–405.
[22] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. 2013. Device-to-device communications with Wi-Fi Direct: Overview and experimentation. *IEEE Wireless Commun.* 20, 3 (June 2013), 96–104. DOI: http://dx.doi.org/10.1109/MWC.2013.6549288
[23] M. W. Gielen. 2012. Ad hoc networking using wi-fi during natural disasters: Overview and improvements. In *Proceedings of the TScIT*, Vol. 17.
[24] Kyoung-Hak Jung, Yuepeng Qi, Chansu Yu, and Young-Joo Suh. 2014. Energy efficient Wifi tethering on a smartphone. In *Proceedings of the INFOCOM*. 1357–1365. DOI: http://dx.doi.org/10.1109/INFOCOM.2014.6848069
[25] Kyoung-Hak Jung, Jae-Pil Jeong, and Young-Joo Suh. 2014. Sleeping mobile AP: A novel energy efficient Wifi tethering scheme. *Wireless Netw.* (2014), 1–18. DOI: http://dx.doi.org/10.1007/s11276-014-0798-7
[26] Hao Han, Yunxin Liu, Guobin Shen, Yongguang Zhang, and Qun Li. 2012. DozyAP: Power-efficient Wi-Fi tethering. In *Proceedings of the MobiSys*. 421–434.
[27] Ashish Sharma, Vishnu Navda, Ramachandran Ramjee, Venkata N. Padmanabhan, and Elizabeth M. Belding. 2009. Cool-tether: Energy efficient on-the-fly wifi hot-spots using mobile phones. In *Proceedings of the CoNEXT*. 109–120. DOI: http://dx.doi.org/10.1145/1658939.1658952
[28] Sacha Trifunovic, Bernhard Distl, Dominik Schatzmann, and Franck Legendre. 2011. WiFi-Opp: Ad-hoc-less opportunistic networking. In *Proceedings of the ACM CHANTS*. 37–42. DOI: http://dx.doi.org/10.1145/2030652.2030664
[29] Felix Busching, Sebastian Schildt, and Lars Wolf. 2012. DroidCluster: Toward smartphone cluster computing—The streets are paved with potential computer clusters. In *Proceedings of the ICDCS Workshops*. 114–117.
[30] Sacha Trifunovic, Andreea Picu, Theus Hossmann, and Karin Anna Hummel. 2013. Slicing the battery pie: Fair and efficient energy usage in device-to-device communication via role switching. In *Proceedings of the 8th ACM MobiCom Workshop on Challenged Networks*. 31–36. DOI: http://dx.doi.org/10.1145/2505494.2505496
[31] Hanno Wirtz, Tobias Heer, Robert Backhaus, and Klaus Wehrle. 2011. Establishing mobile ad-hoc networks in 802.11 infrastructure mode. In *Proceedings of the ACM CHANTS*. 49–52. DOI: http://dx.doi.org/10.1145/2030652.2030666
[32] Md. Yusuf Sarwar Uddin, Hossein Ahmadi, Tarek F. Abdelzaher, and Robin Kravets. 2013. Intercontact routing for energy constrained disaster response networks. *IEEE Trans. Mob. Comput.* 12, 10 (2013), 1986–1998.
[33] Md. Yusuf Sarwar Uddin, Hossein Ahmadi, Tarek F. Abdelzaher, and Robin Kravets. 2009. A low-energy, multi-copy inter-contact routing protocol for disaster response networks. In *Proceedings of the IEEE SECON*. 1–9.
[34] Zongqing Lu, Guohong Cao, and Thomas F. La Porta. 2016. Networking smartphones for disaster recovery. In *Proceedings of the IEEE PerCom*. 1–9.
[35] M. Raj, K. Kant, and S.K. Das. 2014. E-DARWIN: Energy aware disaster recovery network using wifi tethering. In *Proceedings of the ICCCN*. 1–8. DOI: http://dx.doi.org/10.1109/ICCCN.2014.6911770
[36] Cell On Wheels. [n.d.]. Retrieved from https://cellsitesolutions.com/portfolio-view/cows/.
[37] Naoto Kadowaki, Takashi Takahashi, Maki Akioka, Yoshiyuki Fujino, and Morio Toyoshima. 2012. Research and development issues of satellite communications systems for large scale disaster relief. *IEICE Trans. Commun.* 95, 11 (2012), 3378–3384.
[38] H. Kazerooni. 2006. The berkeley lower extremity exoskeleton project. In *Experimental Robotics IX*. Springer Tracts in Advanced Robotics, Vol. 21. 291–301.
[39] Balaji Raghothaman, Eric Deng, Ravikumar Pragada, Gregory Sternberg, Tao Deng, and Kiran Vanganuru. 2013. Architecture and protocols for LTE-based device to device communication. In *Proceedings of the ICNC*. 895–899.
[40] WiFi direct. [n.d.]. Retrieved from https://www.wi-fi.org/discover-wi-fi/wi-fi-direct.

[41] FEMA. 2011. Integrated Alert and Warning System (IPAWS). Retrieved from https://www.fema.gov/pdf/emergency/ipaws/ipaws_factsheet.pdf.
[42] Guide to Implementing the Integrated Public Alert and Warning System (IPAWS). [n.d.]. Retrieved from https://www.csepportal.net/TrainingDocuments/IPAWS_HowToGuide_21JUL2014.pdf.
[43] Reuven Cohen and Boris Kapchits. 2011. Continuous neighbor discovery in asynchronous sensor networks. *IEEE/ACM Trans. Netw.* 19, 1 (Feb. 2011), 69–79.
[44] Sudarshan Vasudevan, Micah Adler, Dennis Goeckel, and Don Towsley. 2013. Efficient algorithms for neighbor discovery in wireless networks. *IEEE/ACM Trans. Netw.* 21, 1 (Feb. 2013), 69–83.
[45] Chih-Min Chao, Jang-Ping Sheu, and I-Cheng Chou. 2006. An adaptive quorum-based energy conserving protocol for IEEE 802.11 ad hoc networks. *IEEE Trans. Mob. Comput.* 5, 5 (2006), 560–570.
[46] Lin Chen, Ruolin Fan, Kaigui Bian, Lin Chen, Mario Gerla, Tao Wang, and Xiaoming Li. 2015. On heterogeneous neighbor discovery in wireless sensor networks. In *Proceedings of the IEEE INFOCOM*. 693–701.
[47] Chih-Min Chao and Yi-Wei Lee. 2010. A quorum-based energy-saving MAC protocol design for wireless sensor networks. *IEEE Trans. Vehic. Technol.* 59, 2 (2010), 813–822.
[48] Chu Luo, Henri Koski, Mikko Korhonen, Jorge Gonçalves, Theodoros Anagnostopoulos, Shin'ichi Konomi, Simon Klakegg, and Vassilis Kostakos. 2017. Rapid clock synchronisation for ubiquitous sensing services involving multiple smartphones. In *Proceedings of the ACM UbiComp/ISWC*. 476–481.
[49] Power Monitor. [n.d.]. Retrieved from https://www.msoon.com/LabEquipment/PowerMonitor/.
[50] Amitangshu Pal and Krishna Kant. 2018. E-Darwin2: A smartphone-based disaster recovery network using WiFi tethering. In *Proceedings of the IEEE CCNC*. 1–5.

Received March 2019; revised September 2019; accepted November 2019