A Smartphone based Network Architecture for Post-Disaster Operations using WiFi Tethering

Amitangshu Pal*, Mayank Raj†, Krishna Kant*, Sajal K. Das†
† Temple University, Philadelphia, PA
‡ Missouri University of Science and Technology, Rolla, MO
E-mail:{amitangshu.pal@temple.edu, mraj@us.ibm.com, kkant@temple.edu, sdas@mst.edu}

Abstract—Emergency communication networks are crucial for monitoring and providing assistance to affected people during long-persisting disasters such as Tohoku earthquake in Japan or Hurricane Katrina in the US. Given ubiquity of smartphones, we envision future emergency networks to seamlessly integrate the smartphones in the disaster area, unfailed portions of the cellular network, and the emergency communication equipment deployed the area. The paper proposes such an architecture called Energy Aware Disaster Response Network using WiFi Tethering (E-DARWIN). The key innovation in this network includes the use of WiFi tethering technology for smartphones that have lost cellular connectivity since this feature is becoming almost universal on all smartphones. The network uses a central control of the entire network through an emergency control center (ECC) to ensure awareness of smartphone battery state, most appropriate data collection through various sensors of the phone under software control, and ability to handle mobility of smartphones—both those that connect via WiFi tethering (because they have lost cellular connection) and those that still have cellular access and can act as gateways. We analyze the performance of our network using a combination of mathematical modeling, large scale simulations, and a prototype implementation on an Android platform.

Index Terms—Disaster Communications, WiFi Tethering, Ad-hoc network, Grid based quorum

1 INTRODUCTION

The world has lately witnessed several large disaster scenarios that extend over multiple days or longer, e.g., 2011 Tohoku earthquake in Japan, 2005 Hurricane Katrina, and 2011 Hurricane Sandy in the US. Such extended disaster scenarios continue to evolve, often in unexpected ways, and a situational awareness of their evolution is crucial for providing the right kind of emergency response in terms of rescue, health-care, and other necessities. Large disasters often damage the communications infrastructure, and an ad hoc emergency communications infrastructure becomes essential to provide communications coverage. Given the ubiquity of smartphones and a wide variety of sensors available on them to capture critical data (e.g., pictures, sound, vibrations, etc.), they are best positioned to form an ad hoc network through local communications technologies such as WiFi, ultimately connecting to the Internet via Gateway device with both local (e.g., WiFi) and long distance (e.g., satellite or cellular) capabilities. The latter may include the following types of devices: (a) Smartphones that still have a cellular coverage (along with WiFi capability), (b) Satellite phones carried by rescue personnel (again with WiFi capability), (c) WiFi access points (along with cellular, satellite, or other long distance link on the other side) mounted on emergency communications vehicles (ECVs), and (d) Especially designed WiFi AP’s air-dropped strategically. The deployment of AP’s on the ground is hampered by the disaster related damage and that of air-dropped ones by the limited battery life and deployment challenges. It is expected that such a network will evolve dynamically due to movement/changes in both the smartphones and gateways and must be considered in network setup, upkeep, and use.

Smartphone based communications have played a crucial role in several recent disaster such as tracking down people and assessing their needs in several recent disasters, most notably 2010 Haiti earthquake [1]. Our purpose in this paper is to go beyond the people-to-people communications using smartphones that happen to have the coverage. Instead, we wish to build an entire ad hoc network as discussed above to cover largest possible sections of the disaster affected areas and use this network primarily to enable collection of important sensor data from smartphones under the control of an authenticated central disaster management facility, often called Emergency Control Center (ECC).

We assume that every smartphone will have our E-DARWIN app installed if it wishes to participate in the ad hoc network. In order to conserve smartphone battery, a fundamental goal is to keep the app in a “dormant” mode most of the time, springing to life to build the ad hoc network at the onset of damage to the normal cellular network, and then act as an element of disruption tolerant network (DTN). That is, even when the ad hoc network is up, the app instances on most phones are in a dormant state, and waking up periodically to do some data transfers and handling of any changes to the network. Such an operation cannot support streaming media applications (e.g., live voice communications), but can support SMS, email, and transfer of accumulated sensor data. Henceforth we focus on this DTN style data transfer aspect exclusively. Smartphones that have a direct connection to Internet could surely use voice or other applications if they choose to, but we do not consider them here. It is also conceivable to design the
In addition to DTN operation, a crucial technique for minimizing battery drain is to avoid collecting unnecessary or redundant data. One would normally expect a huge variation in the distribution of smartphones in the disaster area. In area with a large number of smartphones, it is unnecessary to collect sensor data from all of them, and the ECC can make judicious choice concerning what sensor data each smartphone will provide to fairly cover an entire area without unnecessary redundancy. In the past we have examined an approach based on distributed coalition formation game to decide contributions from each smartphone, and this mechanism could easily consider the remaining battery in the determination [2]. Therefore, the redundancy minimization aspect is not addressed in this paper.

Integrating a large number of devices into the emergency network poses several challenges, which form the main contributions of this paper. The first contribution concerns network initialization, i.e., building the ad hoc network following the disaster as soon as possible while minimizing unnecessary battery drain on the smartphones. The second contribution is building the ad hoc network using WiFi tethering, which is the only technology that is ubiquitously available on almost all smartphones but poses unique challenges in building an ad hoc network. A related contribution is analytic modeling of discovery which shows that the neighboring devices can be discovered with a very high probability in a few rounds. The third contribution is the grid based allocation of slots for a device to be in client or hotspot state for data transfers. This mechanism is tolerant of mobility, which is an important characteristic of user behavior in a disaster situation. It also allows data transfer rates in an energy efficient manner based on the remaining battery or priority assigned to the device. We show that our mechanism can deliver packets to the ECC with high delivery ratio (> 97%) while meeting the various constraints. Although the paper is based on our prior conference publication [2], it includes many enhancements including detailed analysis network initialization and data transfer mechanisms that are tolerant of smartphone mobility.

The rest of the paper is organized as follows. Section 2 describes the basic architecture of the proposed network, called Energy Aware Disaster Response Network Using WiFi Tethering (E-DARWIN). Sections 3 and Section 4 respectively discuss the network initialization and data forwarding mechanisms, including their analytic modeling. Section 5 presents detailed evaluation of our proposed architecture using both a prototype implemented on Android platform and large-scale simulations. Finally, Section 6 concludes the paper.

2 BACKGROUND AND PROBLEM DESCRIPTION

As discussed above, we assume that every smartphone is equipped with our E-DARWIN app. This app is used for establishing the ad hoc network, collecting data under program control from available sensors (such as camera, microphone, gyroscope, GPS, etc.), and transferring data to the next-level network. The latter consists of the gateway devices mentioned before, including the smartphones with surviving cellular links and the emergency APs. Since the APs are intended exclusively for assisting with emergency network, we can assume that they are pre-authenticated by the ECC and have a mechanism to establish connection to the ECC. In addition, all smartphones are assumed to be capable reaching the ECC via a well known URL. The AP interface to ECC could use emerging technologies such as 802.22 or 802.11ah, in addition to the more traditional technologies such as Satellite or cellular [3]. We assume that ECC is established by a public aid agency (e.g., Federal Emergency Management Agency (FEMA) in the US) in some secure location for providing disaster relief, and is a permanent entity that only needs to be fired up whenever necessary. We further assume that 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 and determines data collection needs, collects and analyzes all data, instructs network reconfiguration, etc.

The entire architecture is shown in Figure 1. Although there are many important issues to address for the next level network (e.g., positioning of ECVs, choice of communications technologies, etc.), our focus in this paper is mostly on the smartphone end of the network. We assume that all smartphones have the Global Positioning System (GPS) capability and are routinely time synchronized using the Network Time Protocol (NTP). It is possible to maintain synchronization even following the disaster [4,5], but we do not focus on this aspect.

There are many technologies (LTE direct, WiFi Direct, Ad hoc mode WiFi etc.) that can support peer to peer communications between smartphones, however, they are (a) either not widely supported in current smartphones, (b) or require root access on the device, (c) or require an intermediary (cell tower) for discovery and connection establishment, which is problematic in disaster scenarios. In view of this, we exploit WiFi Tethering (WiFi-T) as the key communications technology for building the ad hoc network. WiFi-T is ubiquitously supported by all major device manufacturers and is available across all major OS platforms, such as iOS 4.3+, Android 2.2+ and Windows 7.5+. It allows a device to act as a WiFi hotspot, while allowing other devices in its communication range to connect to it. The use of WiFi-T requires a device to alternate between client and hotspot modes to forward data (since a link can exist only between a client and a hotspot). This essentially results in a DTN with most links established selectively.
and intermittently as needed, which is also crucial from the perspective of minimizing drain on smartphone batteries. The key challenges in the mechanism include mutual device discovery/initialization, building the entire network with ECC at its apex, efficiently collecting data, and delivering it to the ECC with suitable considerations of smartphone battery drain, available bandwidth, reachability, data transfer latency requirements, and the exploitation of redundancy in collected data to reduce battery drain.

2.1 WiFi Tethering Based Networking

The smartphone network operates in two distinct phases, henceforth called Initialization phase and Data Forwarding phase. We assume that all devices have the E-DARWIN app installed and enabled. When triggered via an emergency trigger (discussed later), the app on each device goes into Initialization phase where it discovers neighbors and builds an ad hoc network, and ultimately either confirms the emergency and goes into the Data Forwarding phase, or simply folds (no real emergency). In the Data Forwarding phase, each device adapts its duty cycle and transmission schedules based on their available battery capacities and data transmission needs, and forwards sampled data periodically. The two phases are discussed in detail in sections 3 and 4.

The WiFi tethering based operation require that each device be in one of the following three states:

Dormant - A device by default stays in the dormant state to conserve energy, until it is scheduled to act either as a WiFi hotspot or client. In this state, all the network interfaces of the device are disabled and the entire device is in a low power mode with the CPU sleeping. Note that this state is different from the dormant state in the medium access layer, wherein 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, so that any neighboring clients can associate with it and offload data to it. Devices act as hotspot for a predetermined time interval and then enter the dormant or client state.

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 the selection of the relay device and offloading of data to it. A device stays in the client state as long as it has packets to offload, and accordingly enters the dormant or hotspot state.

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

Fig. 2 illustrates the typical behavior of the devices in E-DARWIN following the initialization phase. Each device $m, \forall m \in \{1, 2, \ldots, 8\}$ is represented as $D_m$. Assume that at $t = 25$ mins $D_3$ is scheduled to act as a WiFi hotspot. Devices $D_1$, $D_2$ and $D_4$, which are in the communication range of $D_3$ wake up as WiFi client at $t = 25$ mins and associate with it. Thus at $t = 25$ mins, $D_1$ and $D_2$ can forward their stored data packets to $D_3$, which can forward them 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 and then become dormant.

At $t = 27$ mins, $D_3$ and $D_6$ are scheduled to wake up in hotspot and client mode respectively. $D_4$ goes to 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 range of the 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.

2.2 Related Work

Using sensors for constructing disaster recovery network and capturing data has been extensively explored in literature [6]–[9]. However, making use of wireless devices, like smartphones, PDAs and tablets, widely available among people in the affected region is a more practical approach [10,11]. Even though the solutions discussed in literature utilize the ad hoc mode of operation in wireless devices to construct disaster recovery network [12]–[15], implementing it on current wireless devices requires modifications of the transceiver driver firmware, root access to kernel or jailbroken devices [16,17]. Evidently, this is undesirable, as these solutions cannot be seamlessly supported across all devices and are illegal in many countries. This motivated us to explore the use of technologies available on the wireless devices, i.e., WiFi Direct and WiFi Tethering, for constructing the disaster recovery network.

WiFi Direct or WiFi Peer-to-Peer (P2P) is a standard, which allows devices to communicate with one another...
without the need of a wireless AP [18]. Wi-Fi Direct works by embedding a limited wireless AP in the devices and a capability to “pair” two such devices. WiFi Direct based ad hoc networking for disaster communications has been explored in [19]. However, WiFi direct capability is not widely available and such devices are typically able to pair with only one device at a time. Furthermore, the technology is more energy-hungry than WiFi Tethering (WiFi-T) [20]. Therefore, we selected WiFi-T as the preferred technology for our purposes, although much of the work applies to other technologies as well.

In [21]–[24], the authors study the energy consumption of a device using WiFi-T and propose mechanisms to improve its energy efficiency. Having established the viability of using the technology on mobile devices, researchers have explored its use for communicating with other devices in an opportunistic network environment [25] and creating computing cluster grids [26]. A role based architecture is adopted in such solutions wherein devices take up the role of hotspot or client to set up a mobile ad hoc network [20,27]. The authors in [27], propose the use of network virtualization to let devices simultaneously assume the roles of hotspot and client. However, virtualization of the network interfaces is not supported in current mobile devices. Furthermore, the authors in [20] and [27], do not address issues related to network formation, routing and maintenance. To the best of our knowledge, no other work in literature investigates the construction of an ad hoc network for disaster scenarios using WiFi-T and routing of data using it. A preliminary version of our work with a different data forwarding scheme was published in [2]. The data forwarding scheme considered here is based on grid quorum mechanism, and is designed to easily handle the mobility of the smartphones. This earlier paper also presented a game theoretic mechanism to exploit the redundancy in data so that the data transmissions (and hence the battery drain) can be reduced. While this mechanism can be used here as well, we do not focus on the redundancy aspect in this paper.

3 INITIALIZATION PHASE

3.1 Activation of E-DARWIN Application

The initialization of E-DARWIN network needs to address two coupled problems: (a) WiFi-tethering based discovery of neighbors to build locally connected clusters of smartphones, each of which can reach the Internet via one or more gateway devices, and (b) to ensure, with minimum energy expenditure, that there is a real emergency that this network is going to serve. The problems are coupled because (b) requires (a), and doing (a) without (b) is wasteful and highly undesirable.

To resolve this dilemma in the US context, we link E-DARWIN to official emergency alert system, such as Wireless Emergency Alert (WEA) system in US. Fig. 3 shows the current systems in US, with similar systems elsewhere that also use the Common Alerting Protocol (CAP) to source alerts from official sources for driving the open-source IPAWS (Integrated Public Alert and Warning System), that integrates various specific subsystems including WEA involving the cellular carriers [28]. Currently, all 4 major cellular providers in the US and many smaller ones implement WEA and nearly all Android/IPhone phones support it. It is also worth noting that WEA messages ride the control channel and thus are not affected by network congestion.

We assume that the reception of type-2 WEA alert (imminent threats to safety) can start the E-DARWIN app. The first thing the app does is to capture GPS location of the phone and correlate it with the affected area specified in the WEA alert and determines if it is vulnerable, i.e., located in the affected zone. (In this paper, we do not go into the details of how this processing is done.) If the phone does not find itself vulnerable, it simply shuts down the E-DARWIN app. We assume for simplicity that such phones do not have any further role to play unless the app is started manually and instructed to participate. The significance of this assumption is that we do not consider automated handling of non-vulnerable phones later moving into the affected area (except by deliberate choice, in which case a manual start of the app is required).

After the E-DARWIN app determines that it is vulnerable, ideally it should not start building the WiFi based network until it actually loses the cellular connection. Unfortunately, doing so will mean that no phone that remains cellular connected will be able to join in and act as a gateway. Thus, we need a somewhat different procedure for Affected Vulnerable Devices (AVDs), i.e., devices that have actually lost cellular connection, and Unaffected Vulnerable Devices (UVDs), i.e. devices that do not lose cellular connection. Note that the devices can use their phone numbers or MAC addresses as their IDs in the discovery process, and it is reasonable to assume that each device maintains fairly accurate time (down to a few seconds). A vulnerable device that has lost its cellular connection continuously for some fixed time (say, 15 minutes) classifies itself as AVD and starts the discovery process immediately. If it later regains cellular connectivity for some fixed time, it can reclassify itself as UVD. We normally expect AVD to UVD transition to be very infrequent, and thus it suffices to have a completely separate mechanism for UVDs, which we discuss later.

As the discovery proceeds among the AVDs and the WiFi tethering network expands, the network the discovery process ultimately includes phones that have cellular access and the emergency APs deployed in the disaster area. In reality, the deployment of emergency APs will take at least a few hours (and possibly much longer) following the loss/degradation of the cellular network. Thus it is necessary to fold an already built cluster of nodes that cannot reach ECC; yet, it should be possible to retry when new emergency APs are eventually deployed or cellular access becomes at least partially functional.
It is worth noting that the cellular connectivity typically does not go down over a large area suddenly, and is more likely to degrade gradually over hours or longer. Thus it is possible that in the initial stages of disaster impact, there are enough UVDs in any area so 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 get entirely isolated some time, and then perhaps able to reach ECC again through APs, as they come online. In any case, a full coverage is not possible in general, and our concern is to use the AVDs that can reach ECC for data transfer while continuing to discover potential new pathways. The discovery process needs to be extremely lightweight so that the battery drain on useless cluster buildouts can be minimized. The detailed discovery process is described in section 3.2.

In this work we primarily focus on automated initiation of network buildout since the people carrying the phones may be incapacitated, confused, or otherwise not in the best position to manage this process. It is possible to augment this mechanism with more direct means such as user directed initiation. It is also possible to trigger the process not based on connectivity loss, but on sensors detecting relevant signals themselves. Two examples of the latter are (a) seismic vibrations based detection of earthquake [29]–[31], and (b) monitoring of appropriate emergency sirens [32,33].

We now briefly address the security and privacy issues with E-DARWIN. The biggest vulnerability of the proposed scheme is fraudulent triggering of ad hoc network construction and subsequent data collection, and possibly draining of phone batteries in the process. Tying the network buildout to WEA ensures that a few rogue phones cannot result in network building. Furthermore, we require that each phone authenticates itself immediately after it is able to talk with ECC. The authentication is essential to ensure that the ad hoc network can be trusted before any non-control data flows through it. We assume that the emergency APs are pre-authenticated by the ECC, and thus ECC and APs form the root of the trust. This can be propagated down to other devices in the cluster using well known mechanisms, such as public-key-infrastructure (PKI) based digital signatures [34] or group signature based security schemes [35,36]. ECC/AP may also ask credentials of the devices to ensure that rogue devices are not included in the emergency network. Since any device in our network may temporarily lose connectivity to ECC due to movement or signal loss, it may be desirable to consider authentication valid for certain period of time (e.g., for 1-2 hours).

We believe that the above security mechanism are adequate in an emergency scenario. It is certainly possible for a rogue phone to become part of the ad hoc network and falsify data or not forward it properly, however, mechanisms to address such things will perhaps be an overkill. As to the privacy, we follow the standard practice of asking user’s permission about the use of sensors and usage of resources when the app is installed or when the user wishes to modify these parameters. We don’t believe that heavy duty privacy preservation mechanisms would be practical.

### 3.2 Neighbor Discovery

When the E-DARWIN application is activated, the devices need to discover their neighbors and learn their neighbor’s schedule. As discussed in the last section, there are two types of phones that we need to consider during the disaster: AVDs and UVDs. (We assume that if an AVD cannot access Internet even if it was connected to a normal WiFi AP before the interruption.)

We first describe how the AVDs discover one another. Existing solutions in the literature for neighbor discovery rely on coordination between the devices to ensure that their radios are simultaneously turned on for discovering one another [37,38]. With Wifi tethering, they also must have their WiFi radios turned on and in different states, i.e., one acting as WiFi hotspot and the other as WiFi client. To address this problem, the devices are restricted to stay in hotspot state during the initialization phase for $T_{\text{init}}$ time. During the period $T_{\text{init}}$, the devices consecutively take up the role of WiFi client after random time ($C_{\text{next}}$), that is uniformly distributed between $(0, C_{\text{disc}})$. The random role assumption allows the devices to be in different states after random time intervals, and discover one another.

During initialization, a device in the hotspot state waits for one of its neighbors to become client and associate with it. We assume that time is divided into slots with a duration of $I_{\text{disc}}$. When the device becomes 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 one another to the neighbor list and exchange one another’s schedule, which is described in section 4.

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 be continuously trying to discover their neighbors. From the perspective of E-DARWIN, a UVD will turn on its WiFi radio in client mode for short periods so that it can discover any nearby AVDs that are in hotspot mode. The wake up frequency can be either periodic or adaptive. In 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 increases it exponentially for the next round up to a maximum threshold $Z$ (like 1 hour). If any new AVDs are found, the sleep duration is reduced to the starting value. Such a discovery mechanism ensures that if there are no AVD’s in the neighborhood of a cluster of UVDs, they will not form any network because all of them will be in client mode. At the same time, any AVDs will be contacted and 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 above mechanism becomes somewhat challenging if the UVD is accessing the Internet as a WiFi client to some local WiFi AP, as is often the case. It is reasonable to assume that if the UVD retains direct cellular access, it will also retain Internet access via the WiFi AP (if it doesn’t, we no longer have an issue). In this case, the UVD will be required to periodically break its association with the normal AP
so that it can connect to 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 access without any user intervention. Furthermore, the switchover 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, not with one of the AVDs that happen to be in the hotspot mode.

During discover, each node in a cluster being built keeps track of its status, which can take five values: NC (Not Connected, which is the initial status), LC (Locally Connected, when it has some connected neighbors), GC (Globally connected via a gateway), and GCA (GC and authenticated by ECC). When a node first connects directly to ECC, it immediately goes through the authentication (discussed in next paragraph), and then changes its status to GCA. It then broadcasts its reachability to other nodes, which change to GC status and finally to GCA (after the authentication by ECC). Thus for any ad-hoc cluster that is built, either all the nodes ultimately go into GCA state, or stay in LC state. Now, if a node stays in LC mode for a time period $T_{on}$, it drops off for a period $T_{off}$ and then tries to rebuild the cluster. Note that while $T_{on}$ can be rather short (a few minutes), $T_{off}$ can be set to an hour or more (perhaps even increasing on successive tries), so as to minimize battery drain. The retry is stopped on restoration of normal connectivity and can otherwise be programmed to stop after a day or so.

Notice that if there are very few UVDs in a disaster-hit region, they may be overloaded for forwarding the traffic of AVDs and exhaust their battery power. We assume that when the infrastructure damage begins, there are plenty of UVDs, and by the time their number has gone down critically, there are deployments of emergency APs to handle the load. The selection of UVDs to work as gateways is controlled by the ECC, i.e. the ECC recruits them depending on their density in any given area, number of APs in the area, traffic needs, etc. The ECC may also use some rotation policies among the available UVDs to utilize their battery power intelligently. Due to mobility, some UVDs acting as gateways may move out of their local region or even lose cellular connectivity. To cope with this, the ECC needs to dynamically recruit new UVDs based on (a) which UVDs have ceased to be the gateway due to mobility, (b) changing traffic needs, and (c) how much bandwidth/battery power a UVD is willing to allocate for transmission. The ECC can also start a bidding game [39,40] among the UVDs to decide how much bandwidth or battery power the devices (or users) are willing to allocate, in exchange of some future rewards such as extra bandwidth by the cellular carrier. To keep the burden on the gateways low, we put a cap on the maximum data rate that an individual AVD can use. More sophisticated load balancing and admission control mechanisms [41,42] can also be considered in this context, which are beyond the scope of this paper.

<table>
<thead>
<tr>
<th>TABLE 1: Table of Notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{init}$ ≜ Time duration of the initialization phase</td>
</tr>
<tr>
<td>$C_{next}$ ∈ 0, $C_{disc}$ ≜ Random time after which a AVD goes to client mode, during the initialization phase</td>
</tr>
<tr>
<td>$I_{disc}$ ≜ Duration of a slot time, during the discovery phase</td>
</tr>
<tr>
<td>$X$ ≜ Discrete time slot corresponding to a continuous time $X$</td>
</tr>
<tr>
<td>$\rho$ ≜ Number of times a AVD goes on client state during the initialization phase</td>
</tr>
<tr>
<td>$T$ ≜ Maximum sleep duration of a UVD</td>
</tr>
<tr>
<td>$s$ ≜ Maximum sleep duration of a UVD, in number of slots</td>
</tr>
<tr>
<td>$N$ ≜ Number of UVD neighbors of a AVD</td>
</tr>
<tr>
<td>$(R_w, C_w)$ ≜ Row and column chosen by device $a$</td>
</tr>
</tbody>
</table>

### 3.3 Analytical Model

#### 3.3.1 Probability of discovery of AVDs

In this section, we present an analytic model to study the impact of the algorithm parameters on the probability of a AVD discovering its neighbors. The necessary parameters are enlisted in Table 1. We divide the time into discrete “slots” with duration 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. Henceforth we denote the slot time as $I_{disc}$.

Let $A$ and $B$ denote two devices in initialization phase. As $A$ (or for that matter $B$) consecutively becomes client during the initialization phase, it will become client at least $\lceil T_{init}/C_{disc} \rceil$ times. The probability that $A$ will become client $\rho > \lceil T_{init}/C_{disc} \rceil$ times in the initialization phase can be calculated by modeling it as a renewal process.

Consider a renewal system in which $A$ repeatedly becomes client after random time slots. Let $\{C_i : 1 \leq i \leq \rho\}$ be a sequence of uniformly distributed non-negative random variables, representing the random time slots in which $A$ becomes client during initialization. When a device becomes client, it selects the next random time slot in which it will become client. The probability that $A$ will select a random time slot $C_i = C_{next}$ from the next $C_{disc}$ time slots to become client $i^{th}$ time during the initialization phase is given by

$$f(t) = 1/C_{disc}, \quad C_i \in [1,C_{disc}]$$

The time to the $\rho^{th}$ instance when $A$ becomes client can be calculated as the sum of independent and uniformly distributed random variables $\{S_i\}$, where $S_\rho = \sum_{i=1}^\rho C_i$. The probability density function $f_\rho(t)$ of $S_\rho$ is related to $f(t)$ as the $\rho$ fold convolution of $f(t)$, i.e., $f_\rho(t) = f(t) \oplus f(t) \oplus \ldots \oplus f(t)$. However, the relationship between $f(t)$ and $f_\rho(t)$ can be expressed in a more simplified way using the $Z$-transform of $f(t)$ and $f_\rho(t)$ as follows

$$f(z) = Z[f(t)](z) = z/\hat{C}_{disc}(z - 1)$$

$$f_\rho(z) = Z[f_\rho(t)](z) = (z^\rho/\hat{C}_{disc})^\rho (z - 1)^\rho$$

Using eqn (3), the cumulative probability density function $F_\rho(t)$ for $S_\rho$ in the $Z$-transform domain can be expressed as shown in eqn (4). Taking the inverse $Z$-transform of eqn (4), we get $F_\rho(t)$ as shown in eqn (5).

$$F_\rho(z) = z f_\rho(z)/(z - 1) = z f(z)^\rho/(\hat{C}_{disc})^\rho (z - 1)^{\rho+1}$$
each other, with different UVD and AVD equivalent to \( s \). The probability that \( s \) is discovered by at least one UVD is \( I \), where the sleep interval of the UVD is \( T \). In the initialization phase, we consider the worst case scenario, where the sleep interval of the UVD is \( T \), which is equivalent to \( s = T / I \) slots.

Let us consider two neighbors \( A \) and \( B \) where is \( A \) is UVD and \( B \) is a AVD. 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 \( 1 / \hat{C}_\text{disc} \). Thus at any time slot during the initialization phase, an AVD \( B \) remains in the hotspot mode with a probability \( P_c = (1 - 1 / \hat{C}_\text{disc}) \).

We now calculate the probability that \( B \) wakes up within the initialization phase of \( A \), using Fig. 5. Assume that at time instance 0, \( B \) starts its initialization phase with \( \hat{T}_\text{init} = 5 \) slots. In the worst case scenario, a UVD \( A \) wakes up once in every \( s \) slots, which is assumed to be 8 in Fig. 5. We assume that at any time instance a UVD stays in any one of the \( s \) states, which is defined as the number of slots after which \( A \) will wake up. For example in Fig. 5, we assume \( s = 8 \); i.e. at time instance 0 if \( A \) is in state \( S0 \) then it wakes up at slot 0, whereas if it is in \( S7 \), it wakes up after 7 slots. Thus at time instance 0 a UVD \( A \) stays in any one of the \( s \) states, i.e. it stays in one of the states with a probability of \( 1 / s \). Then \( A \) wakes up within the initialization phase of \( B \) if it stays in any one of \( \{S0, S1, S2, S3, S4\} \) states at time instance 0, i.e. \( A \) wakes up in any one of the \( \hat{T}_\text{init} \) slots with a probability of \( \min(1, \hat{T}_\text{init} / s) \). Thus the probability that a UVD discovers its neighboring AVD is given by:

\[
P(A & B \text{ in client \& hotspot mode}) = \begin{cases} 
P_c \hat{T}_\text{init} / s & \text{if } P_c \hat{T}_\text{init} < s \\ 1 & \text{Otherwise} \end{cases} \tag{9} \]

If there are \( N \) UVDs in the vicinity of \( B \), then \( B \) will be discovered by at least one of its UVD neighbors with a probability given by:

\[
P(B \text{ is discovered by at least one UVD}) = \begin{cases} 
1 - (1 - P_c \hat{T}_\text{init} / s)^N & \text{if } P_c \hat{T}_\text{init} < s \\ 1 & \text{Otherwise} \end{cases} \tag{10} \]
We assume $I_{\text{init}} = 30$ seconds, and $T = 1$ hour which is equivalent to $s = 120$ slots. We keep $T_{\text{init}} = 4 \times C_{\text{disc}}$, and vary $H_{\text{disc}}$. From Fig. 4(b) we observe that as $C_{\text{disc}}$ increases, $T_{\text{init}}$ increases as well, which improves the probability of discovery in between UVDs and AVDs. When $C_{\text{disc}} = 25$, $T_{\text{init}}$ becomes 100 which makes the probability of discovery almost equal to 1 as observed from eqn (10).

We next assume $C_{\text{disc}} = 10 \times I_{\text{disc}}$ and vary $T_{\text{init}}$, as shown in Fig. 4(c). Similar to Fig. 4(b) the probability of discovery increases with increasing $T_{\text{init}}$, 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 Fig. 4(b)-(c) we can conclude that as far as $T_{\text{init}} \geq 4 \times C_{\text{disc}}$, $C_{\text{disc}} \geq 10$ and $N \geq 10$, the probability that the AVDs are discovered by their UVD neighbors are $\geq 95\%$. Also with $T = 1$ 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 amount of burden on the UVDs in the discovery phase.

4 Data Forwarding Phase

In the data forwarding phase (also called adaptation phase), the devices mostly remain dormant to conserve their battery power, and occasionally go to hotspot (or client) mode for the necessary communications. For this phase, we adopt a quorum based duty-cycling scheme for the devices. We assume that time is divided into slots. Each slot is large enough to accommodate multiple transmissions.

A quorum set represents the time slots when a device must wake up. In the non-quorum time slots, devices are allowed to enter sleep mode for the entire time slot to conserve energy. Because of using quorums, any two devices will wake up and meet each other at certain time slots, i.e. there are always nonempty intersections between any two quorum sets. In a grid-based quorum, one row and one column are picked in a $n \times n$ grid as a quorum set. This concept can be shown in Fig. 6. Device $A$ picked row $R_a$ and column $C_a$ as its quorum, while device $B$ picked row $R_b$ and column $C_b$. There are two intersections between devices $A$ and $B$, one for $R_a$ and $C_b$ and the other for $C_a$ and $R_b$. As mentioned earlier, the devices must wake up at the chosen quorums, which means both devices will wake up at these intersections.

4.1 Grid Allocation principles

The purpose of this grid allocation scheme is to ensure that a device can discover its neighboring devices in spite of the mobility so long as there is always a valid overlap between any devices within a bounded amount of time. A valid overlap in a time slot happens between two devices when one of the devices is in hotspot mode whereas the other one is in 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 than the studied neighbor discovery schemes in sensor networks [43]–[45] mainly because unlike the related works where an overlap ensures a successful discovery, our scheme needs to ensure a valid overlap between two devices to ensure a successful discovery.

First rule: Each row of a grid consists of $n$ consecutive $(\text{mod } n^2)$ slots. For example in Fig. 7, the rows consist of $n = 4$ consecutive slots, i.e., $(1 \ldots 4)$, $(5 \ldots 8)$, $(9 \ldots 12)$, $(13 \ldots 16)$.

Second rule: $\forall m \in [1, 2, \ldots, n^2]$, any $n$ continuous slots $[m, m + 1, \ldots, m + n - 1]$ $(\text{mod } n^2)$ (11) are distributed in $n$ different columns. For example in Fig. 7, with $m = 2$, slots $(2 \ldots 5)$ are distributed to $n$ columns. Using these two simple grid formation principles any neighboring devices have some intersecting slots, which is proven in [43].

Third rule: For WiFi-tethering based communication, it is necessary that the two devices with an intersecting slot be in different modes (hotspot and client) in the intersecting slot. To ensure this, each of the devices has two disjoint quorum sets: one for its hotspot sequence, and another one for its client mode sequence. Two disjoint quorum sets can be easily picked up from the $n \times n$ grid, such that the corresponding rows and columns are different. There will be two slots that are common to both hotspot and client state, which we call common slots (CSs). In such scenarios, the state which represents the row will win. We show the reason in Theorem 4.

For example in Fig. 7, device $A$ chooses (row2, column2) and (row4, column4) as hotspot and client sequence, which results in two common slots 8 and 14 that are chosen as hotspot and client respectively. The cycle of a $n \times n$ grid is $n^2$, which is 16 for $4 \times 4$ grid in Fig. 7. Thus after 16 slots, the device $A$ repeats the same sequence.

A possibility of no valid overlaps: Using these grid formation principles, two devices generally find a valid overlap within a bounded amount of time. For example, in Fig. 7 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 (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 and vice versa, and so no overlap can take place in between them. This
such a scenario happens if (a) these two devices choose exactly the same rows for their hotspot and client states, which happens with a probability of $P_1 = \frac{1}{n(n-1)}$ and $P_2 = \frac{n^2 - 3n + 3}{n(n-1)}$.

Proof 1. Such a scenario happens if (a) these two devices choose exactly the same rows for their hotspot and client states, which happens with a probability of $P_1$, and (b) $A$’s hotspot column does not overlap with $B$’s client column and vice versa, which happens with $P_2$ probability.

We first find out the expression of $P_1$. Notice that out of $n$ rows two devices choose the same rows as a hotspot state with a probability of $\binom{n}{1} \cdot \frac{1}{n} = \frac{1}{n}$. From the remaining rows they choose the same rows as their client states with a probability of $\binom{n-1}{1} \cdot \left(\frac{1}{n-1}\right)^2 = \frac{1}{n-1}$. Thus the expression of $P$ can be expressed as

$$P_1 = \frac{1}{n(n-1)} \quad (12)$$

We now derive the expression for $P_2$. For simplicity, let us assume that $A$ and $B$ choose their column slots in the following sequence: (a) $A$ first chooses its hotspot column, then (b) $B$ chooses its client column, next (c) $A$ chooses its client column, and finally (d) $B$ chooses its hotspot column. First $A$ chooses its hotspot column from any of the grid columns. After that $B$ chooses its client column, which does not overlap with $A$’s hotspot column with a probability of $\frac{2}{n}$. Now $A$ chooses its client column from the remaining $n-1$ columns (except its hotspot column). There are two cases in this scenario: (a) $A$ and $B$’s client columns are identical which happens with a probability of $\frac{n-2}{n(n-1)}$, (b) $A$ and $B$ client columns are different which happens with a probability of $\frac{n-2}{n(n-1)}$. In case (a), irrespective of whatever hotspot column $B$ chooses, it does not overlap with $A$’s client column. In case (b), $B$’s hotspot column does not overlap with $A$’s client column with a probability of $\frac{n-2}{n-1}$. Putting everything together, $P_2$ can be expressed as

$$P_2 = \frac{n-1}{n} \left[ \frac{1}{n(n-1)} + \frac{n-2}{n(n-1)} \cdot \frac{n-2}{n-1} \right] = \frac{n^2 - 3n + 3}{n^2(n-1)} \quad (13)$$

We now claim that $P$ is extremely low even in case of small $n$. Fig. 9 shows the variation of $P$ with different $n$, which validates our analytical expressions against the simulation results. From this figure we can observe that even with $n = 4$, $P$ is less than 5%, and even lower for higher $n$. We thus claim that using these 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 pretty high, the devices generally remain connected even if few device-pairs do not experience valid overlaps.

Theorem 2. Assume that two phones $A$ and $B$ use the identical grid size of $n$. 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 1 does not arise.

Proof 2. As stated earlier, the special case mentioned in Theorem 1 arises if (a) $A$ 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 and vice versa. First assume that condition (a) is not satisfied as $A$ and $B$ choose different rows ($R_a$ and $R_b$ respectively, $R_a \neq R_b$) for their hotspot states. In this case, the client column of $B$ (i.e. $C_b$) will definitely experience a valid overlap with $R_a$, as the overlapping slot is not going to be the CS for $B$. A similar overlap happens in between the hotspot row of $B$ and the client column of $A$. Thus, there will be at least two overlaps within one cycle.

On the other hand 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 $n - 1$ valid overlaps. Thus the theorem is proved.

Lemma 3. Any device goes into client (or hotspot) mode at least once in $2n$ consecutive slots.

Proof 3. The proof is obvious. In a client column there is 2 potential client slots in $2n$ consecutive slots. One of them may be a hotspot slot due to common slot; however, the other one has to be a client slot. Similar logic applies for the hotspot slots as well.

Theorem 4. Using the grid formation rule, any parent-child pair will overlap in a hotspot-client or client-hotspot mode at least twice in $2 \cdot \max(m^2, n^2)$ slots, where $m$ and $n$ are the grid dimensions of the two devices.

Proof 4. First consider the case of homogeneous grid, i.e. $m = n$. In this case by using Theorem 2 we can show that there will be at least 2 overlapping hotspot-client combinations within $m^2$ slots. Now let us consider the case of heterogeneous grid pattern of two devices $A$ and $B$. We assume that $A$ and

Fig. 8: An example when devices $A$ and $B$ do not overlap within a cycle.

Fig. 9: Probability of not overlapping with $n$. 
B use grids dimensions \( m \) and \( n \) respectively. Without loss of generality, we also assume \( m > n \). Also assume that \( A \) has chosen row \( R_a \) and column \( C_a \) for its hotspot mode, and \( B \) has picked row \( R_b \) and column \( C_b \) for its client mode.

Now consider the hotspot slots of \( R_a \) and client slots of \( C_b \). \( R_a \) consists of \( m \) consecutive hotspot slots (due to the third rule). On the other hand in case of \( C_b \), except one slot (assume this as \( s_1 \) which is changed to hotspot as it is a CS) others repeat with an equal difference of \( n \). Now as \( m > n \), there has to be at least one overlap between \( R_a \) and \( C_b \). However as \( s_1 \) is changed to hotspot mode, there may not a situation where there is no valid overlap between \( A \) and \( B \) within \( \Upsilon = \max \{m^2, n^2\} \) slots as shown in Fig. 10.

![Fig. 10: An illustrative example for Theorem 3.](image)

In such a situation, there can be two such scenarios. First, consider the case where \( m^2 \) is not a multiple of \( n^2 \). In such a scenario, \( t \) cannot be the starting point of the cycle of both \( A \) and \( B \) in Fig. 10. In that case \( s_2 \neq s_1 \) and thus there will be a valid overlap at \( s_2 \). Thus there will be at least one valid overlap within \( 2\Upsilon \) slots.

Now consider the case where \( m^2 \) is the multiple of \( n^2 \), i.e. \( m \) can be equal to \( 2n, 3n, 4n \), etc. In these scenarios, \( m \) consecutive hotspot slots of \( A \) will overlap with at least one client slot of \( B \) other than \( s_1 \). Thus there will be more or equal to one valid overlap within \( \Upsilon \) 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.

Notice that for the heterogeneous grid pattern, any two devices have at least two guaranteed valid overlaps within \( 2\Upsilon \) slots as shown in Theorem 3. Only in case of homogeneous grid pattern there is a small probability that two devices do not experience a valid overlap, as claimed in Theorem 1.

The proofs of Theorem 2 and Theorem 4 do not assume any node synchronization. Thus, the neighboring devices still experience valid overlaps within the bounded time almost surely, even in absence of node synchronization.

In general, the grid sizes for different devices can be chosen as either the same or differently. The problem with the former is that it does not distinguish between different remaining battery capacities of the devices. We introduce this distinction by choosing device sleep times based on their remaining battery capacity. Now, 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), 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.

For example, we can categorize the devices into four types (High (H), Medium (M), Low (L), 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. Thus, if the grid size for \( H \) is chosen to be \( h \rightarrow 5 \times 5 \), then \( m, l, v \) (for M, L, VL types) are chosen as follows:

\[
4h - 4/4 = m \rightarrow 7 \times 7 \quad (14) \\
4h - 4/2 = l \rightarrow 10 \times 10 \quad (15) \\
4h - 4/1 = v \rightarrow 20 \times 20 \quad (16)
\]

which ensures 64%, 49%, 36% and 9% active time respectively. By choosing grid size based on each device’s individual battery charge, E-DARWIN solves the fixed listen/sleep frequency problem.

### 4.2 Forwarding Mechanism

The devices transmit frequent beacon messages in their hotspot mode, so that its neighboring devices can associate with it. The devices maintain a neighbor table where they store their neighbors, their path-duty-cycles (path-DCs) (explained later on) along with the time-stamps when it was last associated with that neighbor. If no association happens with a neighbor for a long time (defined as discard interval (DI)), its entry from the table is deleted. The phones also update their neighbor table as well as their path-DCs whenever a new association takes place. The devices also exchange their wake-up schedules (hotspot and client schedules) with their neighbors during the association. The wake-up schedules of the neighbors are utilized in the forwarding phase as mentioned later on. The devices also share whether they are mostly mobile or static in their beacon messages.

**Definition 1.** The path-duty-cycle or path-DC is the product of the DCs of all the devices in a path. A device with high duty-cycle has high energy availability and goes to hotspot and client states more frequently. Thus, the path-DC reflects how good a route is terms of the entire path latency. Also higher path-DC means the intermediate devices have higher battery capacity (as the duty-cycle is proportional to battery-capacity) as vice versa.

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

**Energy-delay aware forwarding:** The path selection is performed based on maximizing the path-DC, i.e. minimize the overall packet latency.

Notice that longer the path is, lesser is the path-DC, i.e. the metric also prefers shorter paths. On the other hand,
the routes always prefer the devices with higher energy availability or shorter $n$ in this process.

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

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

In this process a device chooses a parent with the maximum path-DC value (ties are broken randomly) such that the beacon signal strength (signal strength is used instead of other metrics like ETX to ensure lesser control overhead) from that device is more than some threshold $\Gamma$. In this process the devices effectively form a tree like architecture towards their APs as shown in Fig. 11. Repetitive collisions are avoided by using the binary exponential backoff mechanism of traditional 802.11 standard.

Notice that the wake-up schedules proposed in section 4, 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, in two devices with homogeneous grid dimensions overlap just four out of $n^2$ slots, with $n = 10$ and a slot duration of 1 minute, the devices may experience just 2 valid overlaps (using Theorem 2) in 100 minutes (or one overlap in $\sim$50 minutes). For a multi-hop network, this will result in significantly high latency. On the other hand, reducing the grid dimensions (or increasing the duty-cycles) is not energy-efficient as there are just 2 valid overlaps out of $4n - 4$ active slots. We also believe that the slot duration should not be chosen very small (less than 1 minute) because in WiFi-Tethering the state transitions take $\sim$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, it retries in the successive slots. If a retransmission limit $\Psi$ is reached (i.e. the parent or the device may go out of their range), it decides its next best neighbor as its parent and sends its data packets at the corresponding slots. Fig. 12 shows the average and maximum delay a device needs to wait for an active slot of any neighbor (or parent) with different grid dimensions. From this figure we can observe that with $n = 10$ (i.e. with $\sim$40% duty cycle) the average waiting time is $\sim$2 slots. Such a forwarding policy significantly reduces the end-to-end packet latency and makes the scheme applicable in practical environment for collecting crucial information for situational awareness and at the same time let the devices sleep most of the time. Notice that this energy-delay aware forwarding mechanism is used only by the devices that are mostly static; the mobile devices use an opportunistic forwarding strategy which we mention next.

Hot-potato based forwarding: In the hot-potato based forwarding a mobile device sends its stored packets to any of its static neighbors. To do so, it chooses a grid dimension of $n_{\text{max}} \times n_{\text{max}}$ (which is the maximum grid dimension allowed), and remains on for 2$n_{\text{max}}$ consecutive time slots in client (or hotspot) mode. If it can connect to any neighbor (for multiple hotspot neighbors, one of them will be selected randomly), it uploads its packets to it. Notice that using the grid formation rules, any device goes to its hotspot (or client) mode at least once in 2$n_{\text{max}}$ slots (using lemma 3). Thus in 2$n_{\text{max}}$ 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 still can send its packets successfully to the AP. We strategically choose $n_{\text{max}} = 10$ (we isolate the VL phones from forming a multi-hop tree as they will die fast), thus the mobile devices choose any of the 20 out of 100 slots to remain active in any 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 any of their static neighbors (if any) within bounded delay.

The state transition procedure: In a realistic disaster environment people frequently move depending on their needs, thus 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 it joins the network, otherwise it backs-off and start the rediscovery after certain interval, as shown in Fig. 13. If a device joins and leaves the network very frequently then it can be inferred that the person is in a highly mobile stage. In a mobile state, the device also uses hot-potato based forwarding and discovery process to join the network. If it fails to find any neighbor in this process, 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.
5 PERFORMANCE EVALUATION

We evaluate the performance of E-DARWIN through prototype based experiments as well as through extensive simulations. We notice that there exist designs for communications in disaster recovery, which are broadly in the form of opportunistic, inter-contact based routing. In our simulations, we could consider the rescue vehicles and the police patrol vehicles with long-range radios as potential APs, that travel in different residential areas for humanitarian and surveillance purposes. However, the trajectory of these vehicles are somewhat unexpected and sometimes event driven especially in case of evolving disaster environments.

At the same time when such a vehicle passes by a mobile device, the device may be in sleep mode (as it stays most of the time), thus there may not be any successful inter-contact and message exchange. Thus the inter-contact based routing schemes in the literature [46]–[48] is not appropriate in such an environment. Also in these schemes, communication is possible whenever two devices encounter each other, whereas, in our scheme two devices must be in different modes to ensure a valid overlap. Due to these fundamental differences, we cannot compare them with E-DARWIN.

5.1 Power Consumption Evaluation

For a detailed evaluation, we first implemented the capabilities required by E-DARWIN on a Samsung Galaxy S III device running Android OS 4.4.2. This provided us with energy consumption profiles (using Power Monitor [49]) and also guidelines for selecting various algorithm parameters for the simulations. The results reported in the following are averages over 25 measurements for each experiment.

In the first set of experiments, we measured energy consumption and residence time in each state. As shown in Fig. 14(a), WiFi hotspot is the most energy-intensive 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 active state as it repeatedly broadcasts network discovery beacons, and performs various network operations as well as maintenance, making it the most energy exhaustive state.

We then analyze the impact of various algorithm parameters on the device power consumption in the initialization phase. First, we increase the initialization time (\(T_{\text{init}}\)) while keeping \(C_{\text{disc}} = 10 \times I_{\text{disc}}\) and \(I_{\text{disc}} = 30\) s. As shown in Fig. 14(b), devices consume more power with increasing \(T_{\text{init}}\). The reason is that with increasing \(T_{\text{init}}\), the devices transition in between the hotspot and client mode more frequently, which increase the power consumption. Thus, while a high ratio of \(T_{\text{init}}\) increases the probability of a device discovering its neighbor (as \(\rho \propto \frac{T_{\text{init}}}{C_{\text{disc}}}\)), it results in higher power consumption for the devices. Conversely, increasing \(C_{\text{disc}}\) with constant \(T_{\text{init}}\) (assumed to be \(10^4\) seconds) reduces the power consumption of the devices as they transit in between hotspot and client state less frequently, as shown in Fig. 14(c). For the similar reason, the power consumption decreases with increasing \(I_{\text{disc}}\). But this will also reduce the probability of the two devices discovering one another as there is a higher probability of them repeatedly becoming client at the same time.

5.2 Simulation Based Network Performance Evaluation

We next study the performance of the proposed quorum based data forwarding scheme in Castalia [50], 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 \(\times\) 200 sq. m. In the presence of device mobility, this fixed topology changes based on the movement pattern of the devices, which are mentioned later on.

We assume a log-normal shadowing model with path-loss exponent \(k = 2.4\), and channel variance \(\sigma = 4\) dBm. The path loss at a reference distance \(d_0 = 1\) is assumed to be \(55\) dBm. We assume additive interference model for our simulations. The transmit power is assumed to be 20 dBm, which is the representative of current smartphones. The signal-delivery threshold and Clear Channel Assessment (CCA) threshold are assumed to of -80 dBm and -95 dBm respectively. \(\Gamma\) is assumed to be of -73 dBm to ensure high possibility of successful packet reception at the receiver end.

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 statically located devices and also those where a device
moves in a random direction at a certain speed (nominally 1 m/sec) 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 CSMA/CA as their access protocol, with a maximum retransmission count of 30. We deactivate the RTS/CTS mechanism to conserve the device’s remaining energy. Unless otherwise mentioned, we choose quorum grid-sizes of \( 5 \times 5 \), \( 7 \times 7 \), and \( 10 \times 10 \) 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/secs (one packet in 10-100 seconds) which is sufficient for our application.

**Effect of different packet transmission rates:** Figs. 15a and 15b show the packet delivery ratio and packet delivery latency with the variation of packet transmission rates, which are assumed to be constant for all devices. We can observe a slight increase in the average packet delay while the data rate increases from 0.02 packets/secs to 0.1 packets/secs. 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 the proposed forwarding policy.

**Effect of different quorum slot durations:** Figs. 15c and 16a show the delivery ratio and packet delay with the variation of slot durations. The packet transmission rate is assumed to be 0.01 packets/secs. We can observe that the packet delay increases \(~3\)-6 times and almost linearly with the increase in slot durations. This is because of the increasing delay in between the valid overlaps of the devices. The packet delay also increases by \(~10\)-20 times with the increase in mobility. The packet delivery ratio also remains constant and above 98\% in all these experiments. Notice that 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.

**Effect of different grid lengths:** Figs. 16b and 16c show the delivery ratio and packet delivery delay with the variation of quorum grid sizes, where the grid sizes of the devices are assumed to be identical. The packet transmission rate is assumed to be 0.01 packets/secs, whereas the slot duration is kept as 1 minute. We can observe that the packet delay increases by \(~3\)-5 times as the grid size increases from \( 5 \times 5 \) to \( 10 \times 10 \), especially in presence of device mobility. The reason is because a device’s duty-cycle decreases with the increasing grid dimensions, which results in higher waiting time for forwarding the stored packets to the parents.

6 Conclusion

A novel architecture called E-DARWIN was proposed in this paper for creating the network infrastructure in disaster affected regions using WiFi Tethering. Specifically, novel
mechanisms were discussed using which the devices can autonomously discover their neighbors and forward data to the WiFi AP with minimum delay. The proposed solutions were analyzed through a comprehensive set of experiments using a prototype implemented on Android platform and large-scale simulations. In the future, we plan to address many of the assumptions and limitations of the current scheme. In particular we plan to consider mobile devices with a much wider set of characteristics in terms of available sensors, networking capabilities and storage availability as well as the topology of the network to ensure devices crucial to connectivity do not exhaust their battery quickly. We will also examine mechanisms to ensure the devices conform to the actions formulated by the ECC and verify the veracity of the data supplied as well as build trust in the system.

References


[28] “Guide to implementing the integrated public alert and warning system (ipaws).”


