E-Darwin2: A Smartphone Based Disaster Recovery Network using WiFi Tethering

Amitangshu Pal and Krishna Kant
Computer and Information Sciences, Temple University, Philadelphia, PA 19122
E-mail:{amitangshu.pal,kkant}@temple.edu

Abstract- Emergency communication networks are crucial for monitoring and providing assistance to affected people during long-persisting disasters. Given substantial and increasing penetration of smart-phones throughout the world, we envision future emergency networks to consist of smart-phones in the disaster area, un-failed portions of the cellular network, and the communication capabilities provided by the specially deployed emergency equipment (e.g., fixed and mobile wireless access points deployed on the ground or in air via helicopters, satellite radio interfaces, etc.). We envision the emergency network for bulk data transmission – such as transmitting multiple pictures/sounds captured by the phone to help with rescue/safety assessment while keeping the delay/energy expenditure minimum. With extensive simulations, we show that the proposed scheme forwards the sensed data to the control centers with small latency ($<$ 3 seconds) while keeping the WiFi radios on for less than 1% of time.

I. Introduction

With increasing frequency and intensity of natural disasters (examples include Tohoku earthquake in Japan, 2005 Hurricane Katrina, and 2011 Hurricane Sandy in the US), and increasing impact of all types of disasters on large urban areas, it is important to focus on the key issue of facilitating communications during and after the disaster. The communication networks become stressed in the aftermath of a disaster both due to heavier traffic and potentially reduced capacity due to damage to the infrastructure (e.g., cell towers). Numerous factors such as inaccessibility of the area, lack of electricity, unstable buildings, uncertainties about where the impacted people might be, etc. make it very difficult to set up such a communications network expediously and change its configuration dynamically so that it best serves its purpose.

It is well recognized that the ubiquitous use of smartphones and the availability of an increasing array of sensors on them can be a big boon in this regard. In particular, by making smartphone as an integral part of the emergency network, we automatically solve the twin problems of making contacts with the impacted people and gathering relevant data around them, such as their location, pictures/sounds relating to the ongoing event or the damage caused by the disaster, environmental conditions, health-status, etc. Nevertheless, integrating smartphones into the emergency network and making the resulting network cognizant of the needs of the smartphone owners, their remaining battery level, and the physical situation around them, is quite challenging.

In order to integrate smartphones with the rest of the emergency communication network, we assume that every smartphone is equipped with our emergency networking app that is used for establishing the ad hoc network as required to fill gaps in Internet connectivity, data collection under program control from available sensors (such as camera, microphone, gyroscope, GPS, etc.), data filtering and communication to the deployed emergency communications nodes, and ultimately for changing the network configuration in response to instructions from "higher levels" in the deployed emergency infrastructure. A variety of emergency equipment such as WiFi access points (APs), satellite gateways, replacement cellular base stations, etc. mounted in fixed places or on Emergency Communication Vehicles (ECVs) [1], [2] could be part of the deployed network. We assume that there is an emergency control center (ECC) at the apex of this network which has adequate resources to control the entire network. We assume that ECC does much of the heavy duty data analysis, makes high level decisions (usually with human assistance), and initiates control actions.

A key issue in integrating smart-phones in the emergency network is what technologies and protocols to use in building the ad hoc network. There are many technologies that are either available or under development that can support peer to peer communications between mobile phones, however, most of them suffer from some issues. For example, the LTE technology is not only becoming the dominant 4G technology, but also being extended for peer to peer communication in form of emerging standard called LTE direct for this purpose [3]. However, it is unlikely that this technology will be widely available for next 5 years or so. Moreover, LTE direct does require an intermediary (cell tower) for discovery and connection establishment, which is problematic in disaster scenarios. Nearly all smart-phones support WiFi standard, particularly the infrastructure mode. WiFi Direct or WiFi Peer-to-Peer (P2P) is a newer standard, that allows device to device communication by embedding a limited wireless access point in the devices. Although the use of this technology in disaster scenarios is shown in [4], the technology is not widely available and many devices may not have the capability to connect to multiple hotspots or APs at the same time. Ad hoc mode WiFi is a rather old technology but still not widely supported and often requires root access on the device to set up.

In our earlier works [5], [6], we proposed to use WiFi tethering capabilities that are available on nearly every smartphone currently and built a basic disaster recovery network architecture called *Energy Aware Disaster Recovery Network Using WiFi Tethering (E-DARWIN)*. The idea is to use a loosely synchronized mechanism whereby the mobile phones alternate between client and hotspot mode and thereby allow support both a direct discovery and data transfer without any external AP. However, building WiFi-tethering based network is a bit challenging due to the need for complementary modes (hotspot vs. client) among any two communicating smartphones, and energy conservation by keeping the smartphones in a deep sleep model when not communicating actively. Because

This research was supported by the NSF grant CNS-1461932.

978-1-5386-4790-5/18/$31.00 ©2018 IEEE

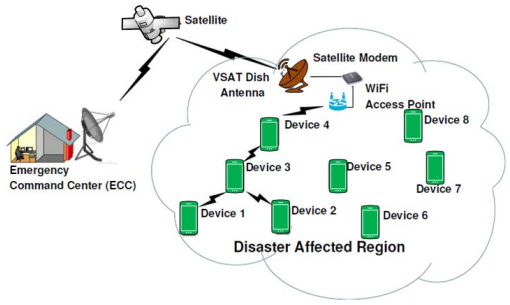

Fig. 1. Network Architecture of E-DARWIN2

of the challenges involved and significant delays (hundreds of seconds) of multi-hop transmissions, the proposed scheme in [5], [6] was suitable for emergency data transfers only, rather than voice communications. In this work we extend this idea and develop a data forwarding scheme (a) within the constraints of acceptable end-to-end data delivery latency suitable for transmitting multiple pictures/sounds packets, (b) minimal storage of packets in the intermediate phones in the data forwarding process to minimize their memory usages, and (c) minimal usage of WiFi radios so that the phones can mostly switch-off them for better energy efficiency. This is achieved by ensuring a better coordination among the devices to remain on at certain time intervals, and transfer a burst of stored packets. The proposed scheme, named E-DARWIN2 ensures small end-to-end latency (< 3 seconds) while keeping the WiFi radios on for less than 1% of time for better energy efficiency.

The rest of the paper is organized as follows. A description of various network components and overall network architecture is presented in section II. The data forwarding mechanisms along with the problem statement is also motivated in this section. As the problem of building multi-hop data forwarding tree with minimum traffic load is NP-complete, we propose a centralized and distributed solutions of this problem in section III and IV. Extensive simulations are carried out in section V. Finally, this paper is concluded in section VI with directions for future work.

II. Network Model and Problem Statement

The network architecture is composed of ECC, WiFi APs with satellite gateway and wireless devices as shown in Fig. 1. The proposed architecture seamlessly integrates wireless devices in a robust ad hoc network, where they behave autonomously, discover and synchronize with one another as well as forward data in a multi-hop fashion to the WiFi AP. Each device in the network can be in only one of the three states at any given instant of time, namely WiFi hotspot, WiFi client and dormant. In Wifi-Hotspot state, the device repeatedly broadcasts network discovery beacons, so that any neighboring clients can associate with it and and offload data to it. In WiFi-Client state, the device periodically scans the wireless channel for advertisements from the hotspot and associates with it.

We assume that the devices discover the presence of an emergency and launches the E-DARWIN2 application autonomously. After launching the E-DARWIN2 application, the phones need to discover their neighbors, and finally join the network in a secured manner, which is detailed in [5]. In this paper we mainly consider the data forwarding and scheduling of the flows in an energy efficient manner. Fig. 2 depicts the functioning of the proposed scheme. In Wifi-tethering, two phones can communicate with each other if they are in their hotspot and client mode respectively. To conserve energy, the phones mostly stay in their dormant (or sleep) state. The devices periodically go to active phase, data transmission phase and low-power phase (or sleep phase). We define each period as an *interval*. In the active phase they all switch on their radio, and randomly choose to remain in hotspot and client mode. In this phase they discover each other and build a multi-hop tree rooted at the ECC. In the data transmission phase, the devices (a) that have packets to send or (b) the parents of those devices remain active during the burst transmission, and then go to sleep. The phones sense different parameters, store them in multiple packets, and forward them towards the ECC in the data transmission phase. Packets belonging to the same flow (or originated from the same source) are defined as *bursts* throughout this paper. The packets belonging to the same burst need to be forwarded in successive transmissions, so that they can be merged together at the control center for effective situational awareness.

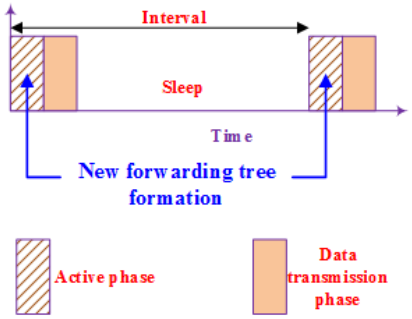

Fig. 2. Timing diagram of E-DARWIN2.

In the active phase, we need to ensure that the multi-hop tree is formed in such a way that it is balanced in terms of the number of *circuits* (or flows) going through the devices. For example Fig. 3(b)-(c) show two possible multi-hop trees that are obtained from the connectivity graph Fig. 3(a), assuming that each device has one burst to transmit. However, in T_1 the maximum number of circuits that will go through any device (device-1 in this case) is 4 (in figure we only show the circuits going through device-1 for clarity), whereas in case of T_2 the maximum number of circuits is 3. Thus T_2 is more balanced as the forwarding tasks are more evenly distributed among the devices. Notice that the active phase and data transmission phase is fairly low (within 1-2 secs) and thus the network topology is assumed to be static during these phases. However, the network topology changes in between different intervals due to the device mobility, thus, the forwarding tree formation needs be done at every interval.

While constructing the multi-hop tree we need to ensure that the maximum number of circuits going through a device remains small. Thus, given a connectivity graph $G = (V, E)$, the objective is to build a multi-hop tree that minimizes the maximum circuits through a device. The problem is identical to the *lifetime maximization problem* [8] in a sensor network where the network lifetime corresponds to the time when the first node in the network runs out of energy. As the lifetime maximization problem is proven to be NP-complete, our problem of minimizing the maximum number of circuits going through a device is NP-complete as well.

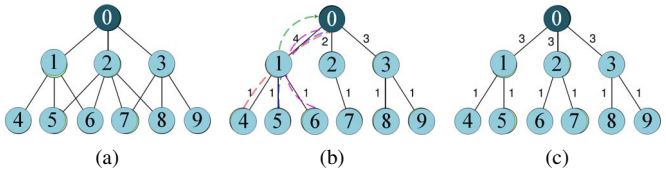

Fig. 3. (a) The connectivity graph. Two types of multi-hop tree formation (b) T_1 and (c) T_2.

III. Centralized Forwarding Tree Formulation and Scheduling

As the problem is NP-complete we propose a centralized heuristic to solve the forwarding free formation, and then propose a centralized interference free schedules to assign the flows at different time instances.

A. Centralized Balanced Tree Formation

We first generate a δ-balanced tree where the difference between minimum and maximum load on the first level nodes is less than δ, similar to [8]. The purpose of building a balanced tree is to avoid the possibility of overloading on certain devices of the network that reduces their remaining battery capacities. We denote $V = \{v_0, v_1, \ldots, v_n\}$ as the set of vertices where v_0 is the ECC, and the first level devices are the APs. Also define the *path load* σ_i of v_i as the maximum load of all the devices along the path from v_i to v_0. For example the path load of (4, 5, 6) in Fig. 3(b) is 4, whereas the path load of (8, 9) is 3. Let v_a and v_b are two leaf nodes of a tree with path loads of σ_a and σ_b respectively. Then for some δ, if $\sigma_a - \sigma_b > \delta$, we say that the tree is in unbalanced condition. Our objective is to build a δ-bounded balanced tree, where $|\sigma_a - \sigma_b| \leq \delta$ for all v_a and v_b. We also define the subtree rooted at node v_i as $T(v_i)$.

We first construct a minimum spanning tree rooted at ECC. If the tree is unbalanced, the node with the highest load (contributing to the maximum path load for some leaf) is selected to initiate the switching of its descendants to different parents to reduce its own load. For example assume that the resultant tree is unbalanced, and the path load of v_a (which is a leaf device) is the highest. Assume that in this tree v_c has the highest load on the paths of v_a and v_x is a descendant of v_c. If v_x has an alternate path to v_0, then it switches with a certain probability (defined as *switching probability*), which reduces the load on the highest path. If v_x does not have an alternate path or is not chosen for switching, the children of v_x are considered in turn.

Such a switching helps in reducing the load on v_c as it does not have to forward packets for the descendants that have switched to alternate paths. But after switching some descendants from $T(v_c)$ to $T(v_d)$, the path load of the leaf nodes of $T(v_d)$ may exceed that of $T(v_c)$ by more than δ. Then the same process is followed to the nodes at $T(v_d)$, where the node with the highest load is switched with some switching probability. This process is continued until (a) the tree becomes balanced, or (b) some node has been selected for switching for a predetermined maximum number of times. If a node switches from one subtree to another, its switching probability is reduced by half to dampen oscillations.

As an example in Fig. 3 assume that δ = 1. Then in Fig. 3(b) $\sigma_1 - \sigma_2 > 1$, thus the tree is unbalanced. As the load of v_1 is the highest, its descendants choose to switch their parent with some switching probability to v_2 to generate a balanced tree shown in Fig. 3(c). Notice that Fig. 3 is constructed assuming that each device has one burst to transmit. However, in general some devices may not have any burst to transmit at any interval. Thus after the tree formation phase, the load of some devices may be equal to 0. These devices (a) neither have any data packet to transmit, nor (b) in the forwarding path of any data packets. Thus these devices can be put to sleep mode during the data transmission phase.

B. Schedule Formation

We next determine the schedules for different flows so that they are interference free. We assume that time is divided into *slots*, and each slot is big enough for a flow to be completed. We also assume that two flows interfere with each others if any of their links are within the interference range of the other. To model the wireless interference among the flows, we first form the *conflict graph* as follows. In the conflict graph, each flow is represented by a vertex. There is an edge in between two vertices (or flows) in the conflict graph, if they interfere, i.e. any link corresponding to the flows are within the interference range of each other.

After forming the conflict graph, we next determine the schedule of the flows, so that the flows that conflict are scheduled at different time. At the same time we need to ensure that the flows are scheduled in the minimum possible time so that the devices that go to sleep the rest of the time. Finding the optimal schedules is NP-hard because of its similarity with the graph coloring problem [9], where two vertices (or flows) can be colored with same colors (or scheduled at the same time) if they do not have a common edge in the conflict graph. Below we describe a simple heuristic solution to this scheduling problem.

To find the schedules, we choose a simple sequential coloring, which proceeds as follows. Assume that the vertices of the conflict graph is ordered as $\vartheta_1, \vartheta_2, \ldots, \vartheta_n$. First the flow corresponding to ϑ_1 colored with c_1. Thus the neighbors of ϑ_1 cannot use c_1. Similarly, each of the remaining vertices ϑ_i in sequence, is assigned the smallest color that, so far has not been assigned to any vertex adjacent to it. After the coloring, the flows with color c_i are scheduled at time slot-i, thus the flows that are scheduled in the same time slots do not interfere.

IV. Distributed Tree Formation and Forwarding Mechanism

We next propose a distributed version of this problem, which consists of distributed tree formation and a burst switching based forwarding mechanism as described below.

A. Distributed Tree Formation

In each interval, the nodes autonomously form a tree which depends on which devices have stored packets to forward to the ECC. To start the process, the APs first broadcast beacon messages with hop-count equal to 0. After receiving the beacon messages, the devices that are in the neighborhood of the APs set their hop-count as 1 and broadcast this in their beacon messages. Assume that h_i is the hop-count of device-i. Then in general, a device-j calculates its hop-count as $1 + h_i$, $\forall$ i in the neighborhood of j. After deciding the hop-counts in this fashion, the devices select the set of devices among their

neighbors as the *potential parents (PPs)*, whose hop-counts are minimum.

The devices then choose their parents as follows. The beacon messages carry a field called *load*, which is given by the number of circuits that a device forwards. While choosing the parents, the devices go to random backoff which is inversely proportional to their hop-counts (this ensures that the leaf devices choose parents ahead of others), and choose the least loaded PP as their parents, and broadcast. The parents rebroadcast their updates loads every time they receive confirmation from their children. After constructing the forwarding tree in this fashion, the devices whose loads are zero can turn off their radios and go to sleep.

B. Burst Switching Based Forwarding

After forming the multi-hop tree, the devices decide their data forwarding schedules in a distributed fashion. The key purpose of this forwarding is to ensure that the packets corresponding to a burst are received within a short time period (i.e. their interarrival time is small). At the same time the scheme needs to ensure that in the forwarding process the intermediate devices (not at the source) storage space is least used, i.e. the waiting time of the packets in the intermediate devices should be minimum. We now develop such a forwarding scheme, which is influenced by the concept of optical burst switching (OBS) [10] in optical networks. OBS is conceptually aimed at the benefits of packet switching but without the need to O-E-O (optical to electrical to optical) at the intermediate nodes. An OBS network consists of nodes that cane be classified into *ingress* and *egress* nodes. An ingress node consists of a router and a burst assembler, whereas an egress node consists of a router and a burst disassembler. Thus an ingress node is responsible for collecting the outside electronic data and aggregate them into bursts according to their destination addresses, i.e. packets that are destined to the *same destination* is aggregated into bursts. In the egress router, the bursts are disassembled back into packets and transmitted to the electronic world.

In E-DARWIN2 application such a burst switching concept is relevant because the scheme needs to ensure that in such multi-hop networks, the intermediate forwarding device's memory is least used. Thus the problem is no different than that in OBS. However, implementing burst forwarding in wireless domain brings special challenges of interference minimization and channel reservation, which are addressed as follows. The devices who have stored packets in their buffer first go into random backoff, if they sense the channel to be free. When the backoff timer expires, they first transmit a control packet *ahead* of the actual burst transmission. The intermediate nodes broadcast a *reserve channel (RC)* message in their 2-hop neighborhood (we assume the interference range to be 2 hops), which ensures that all their 2-hop neighbors abstain from transmitting within the duration of the following burst transmission. After some time Δ, the devices start transmitting the packet bursts. As the channel is reserved for the intermediate devices, the bursts are generally not stored for long time in the intermediate devices. This also ensures that the packets belonging to a burst reach at the APs one after other, i.e. their interarrival time at the APs are minimized. Some intermediate devices occasionally need to store the bursts for some time, if they receive any RC message from any of their neighbors before forwarding the control packet.

For example in Fig. 4 a flow A→E reserves the channel within the neighborhood of the intermediate devices of this flow, thus, all the devices shown in orange abstain from transmission during the time of A→E. However there is one problem of this scheme. When A finds a free channel, it sends a control packet to reserve a channel in it's 2-hop neighborhood. Let us assume that G is in the neighborhood of A, thus, G cannot transmit during the transmission time of A. At the same time, if F finds the channel to be free (assume that F is not in the neighborhood of A), it sends its control packet and reserve B, which is in the neighborhood of F. This results in a deadlock situation, where the packets of both the flows are stuck at B and F. To alleviate this problem, any device that has some outgoing packets first waits for the channel to be free. If it senses a free channel, it transmits the control packet with a probability p. Thus, a device in F→E reserves an intermediate device in A→E only with some probability, which reduces the chances of such deadlock like situation. Also a device cannot reserve a channel for more than a threshold, which forces the two flows to break the deadlock and starts the data forwarding process.

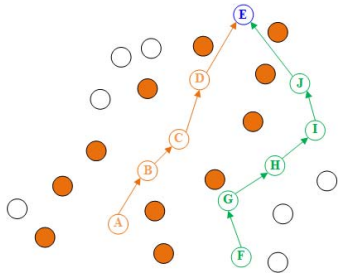

Fig. 4. Burst switching based forwarding.

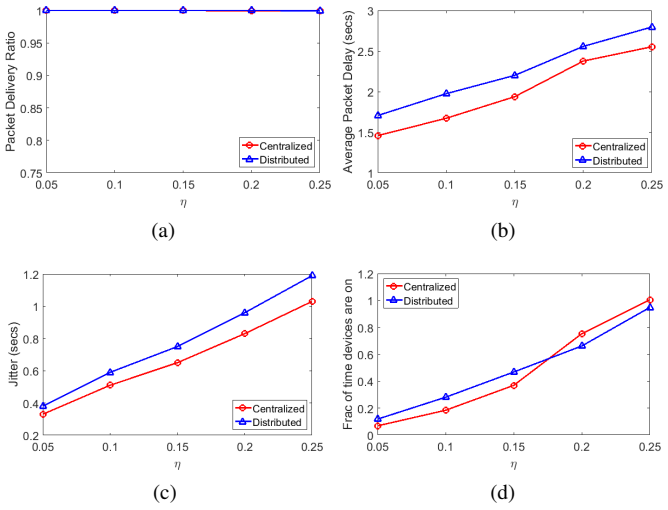

Fig. 5. Comparison of (a) packet delivery ratio, (b) packet delay, (c) jitter, and (d) fraction of time the devices remain active during the forwarding process.

V. Performance Evaluation

We evaluate the performance of E-DARWIN2 through extensive simulations in *Castalia* [11], which is an application-level simulator for wireless sensor networks based on OMNeT++. The simulated system topology consists of 100 devices, placed uniformly in an area of 200×200 sq. m.

We assume a log-normal shadowing model with path-loss exponent k = 2.4, and channel variance σ = 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 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.

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 assume that the devices adopt CSMA/CA as their access protocol, with a maximum retransmission count of 10. We deactivate the RTS/CTS mechanism to conserve the device's remaining energy. Each interval is equal to 100 secs. In each interval, the devices generate a burst of packets with a probability of η. The number of packets in a burst is uniformly distributed within 1 and 10. p is kept to be 0.1 throughout the simulation.

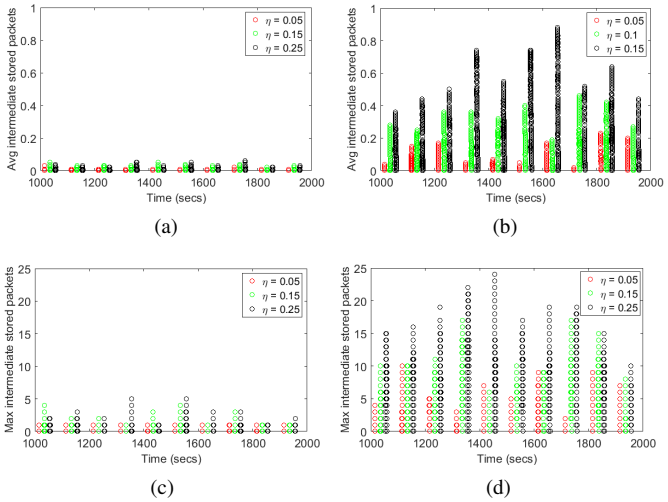

Fig. 6. Comparison of the average and maximum number of packets stored in the intermediate nodes for centralized (a)-(c) and distributed schemes (b)-(d).

From Fig. 5(a) we can observe that the delivery ratio of both the schemes are almost 100% in case of such low-data rate scenarios. Fig. 5(b) shows the average packet latency of the centralized and distributed solution, which is less than 3 seconds even in case of high η. The average packet latency of the distributed scheme is 17% higher than the centralized counterpart. This extra delay is mainly due to the deadlock resolution and the time for the accessing the channel distributedly. Fig. 5(c) shows the comparison of packet jitter (standard deviation of the packet delay). The jitter is slightly more for the distributed solution, because of the more chances of deadlock like situation explained in section IV.

Fig. 5(d) shows the percentage of active time with different η. From this figure we can see that the percentage of active time of both centralized and distributed solution is less than 1%. Notice that the active time of the distributed solution is slightly lower then its centralized counterpart especially in case of high traffic load. Let us explain this with Fig. 4. Assume that two flows A→E and F→E need to be scheduled and link C-D and I-J are in the interfering range of each other. Thus the centralized solution schedules these two flows one after other, whereas in case of the distributed solution the flows are simultaneously scheduled until the packets reach at C and I. After that one of the flows wait for the other one to finish. Because of this parallel transmission, some of the nodes (A-B and F-G in this case) can go to sleep much earlier than others, which reduces the overall active time. Such parallel transmission can also be implemented by designing sophisticated centralized scheme, however, it naturally comes into picture in our distributed solution.

Fig. 6(a)-(b) show the average number of packets that are stored at the intermediate nodes with different η. From these figures we can observe that on average the number of intermediate stored packets is less than 1. In case of centralized scheme, the amount of intermediate storage is almost zero. However, in distributed solution some packets are stored for sometime in the intermediate nodes due to the channel access latencies and deadlock resolution. This clearly shows the effectiveness of E-DARWIN2 for effectively transmitting the packets without involving much intermediate storage. Fig. 6(c)-(d) show the maximum number of packets stored at the intermediate devices, which are limited to 5 and 25 for the centralized and distributed solutions respectively.

VI. Conclusions

In this paper we propose a novel network architecture in disaster affected regions using smartphone based WiFi Tethering scheme. We propose effective mechanisms where the devices can autonomously discover their neighbors and forward data to the WiFi APs. The proposed scheme E-DARWIN2 ensures that the packets are routed with acceptable end-to-end latency and delivery ratio. The scheme also ensures that the devices can sleep most of the time to save their remaining battery capacities. At the same time the memory usage of the intermediate devices in the forwarding process is least used. In the future, we plan to develop a prototype setup to validate the feasibility of the proposed scheme in a real testbed. We also plan to study and address various trust issues that may arise in such a dynamic disaster environment. In particular, the proposed scheme needs to be examined in much more depth in order to determine the robustness of the method and its usefulness in real disaster scenarios.

References

[1] K. T.Morrison, "Rapidly recovering from the catastrophic loss of a major telecommunications office," *IEEE Communications Magazine*, vol. 49, pp. 28–35, 2011.

[2] J. C.Oberg *et al.*, "Disasters will happen - are you ready?" *IEEE Communications Magazine*, vol. 49, pp. 36–42, 2011.

[3] B.Raghothaman *et al.*, "Architecture and protocols for lte-based device to device communication," in *ICNC*, 2013, pp. 895–899.

[4] M.Gielen, "Ad hoc networking using wi-fi during natural disasters: Overview and improvements," in *TSConIT*, 2012.

[5] M.Raj *et al.*, "E-DARWIN: energy aware disaster recovery network using wifi tethering," in *IEEE ICCCN*, 2014, pp. 1–8.

[6] A.Pal *et al.*, "A smartphone based network architecture for post-disaster operations using wifi tethering," in submission. [Online]. Available: https://astro.temple.edu/ tuf79640/TETC_paper-v4.pdf

[7] C.Buragohain *et al.*, "Power aware routing for sensor databases," in *IEEE INFOCOM*, 2005, pp. 1747–1757.

[8] S. K. A.Imon *et al.*, "Rasmalai: A randomized switching algorithm for maximizing lifetime in tree-based wireless sensor networks," in *IEEE INFOCOM*, 2013, pp. 2913–2921.

[9] M. R.Garey and D. S.Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[10] C.Qiao and M.Yoo, "Optical burst switching (OBS) - a new paradigm for an optical internet," *J. High Speed Networks*, vol. 8, pp. 69–84, 1999.

[11] "Castalia: A Simulator for WSN," castalia.npc.nicta.com.au/.