Networking Smartphones for Disaster Recovery

Zongqing Lu, Guohong Cao and Thomas La Porta
The Pennsylvania State University
{zongqing, gcao, tlp}@cse.psu.edu

***Abstract*—In this paper, we investigate how to network smartphones for providing communications in disaster recovery. By bridging the gaps among different kinds of wireless networks, we have designed and implemented a system called TeamPhone, which provides smartphones the capabilities of communications in disaster recovery. Specifically, TeamPhone consists of two components: a messaging system and a self-rescue system. The messaging system integrates cellular networking, ad-hoc networking and opportunistic networking seamlessly, and enables communications among rescue workers. The self-rescue system energy-efficiently groups the smartphones of trapped survivor and sends out emergency messages so as to assist rescue operations. We have implemented TeamPhone as a prototype application on the Android platform and deployed it on off-the-shelf smartphones. Experiment results show that TeamPhone can properly fulfill communication requirements and greatly facilitate rescue operations in disaster recovery.**

I. Introduction

As learned from the 2011 Great East Japan earthquake, the only helpful services in disaster recovery are those that are used daily by everyone [9]. To provide communications in disaster recovery, smartphones, equipped with both cellular and short-range radios (*e.g.*, WiFi, Bluetooth), are the most promising communication tools. Although cellular towers might also be destroyed by disasters, *e.g.*, in the 2008 Sichuan earthquake [3], short-range radios of smartphones can still provide communications. Moreover, the ubiquity of smartphones further opens great opportunities to reinvestigate disaster recovery from the network point of view.

In disaster recovery, smartphones have the potential to be the most feasible communication tools. For example, trapped survivors of a structural collapse can communicate with rescue workers and report their position information through the short-range radio (*e.g.,* WiFi) of their smartphones when they are within the communication range of each other. Smartphones of the rescue workers can also form networks using WiFi and fulfill the communication demand in disaster recovery.

To this end, in this paper, we propose TeamPhone, a platform for communications in disaster recovery, where smartphones are teamed up and work together to provide data communications. By exploiting WiFi and cellular modules of smartphones, TeamPhone seamlessly integrates cellular networking, ad-hoc networking and opportunistic networking, and supports data communications among rescue workers in infrastructure-constrained and infrastructure-less scenarios.

This work was supported in part by Network Science CTA under grant W911NF-09-2-0053 and CERDEC via contract W911NF-09-D-0006.

TeamPhone also enables energy-efficient methods for trapped survivors to discover rescue workers and send out emergency messages, by carefully addressing the wake-up scheduling of smartphones. We implement TeamPhone as an app on the Android platform and deploy it on off-the-shelf smartphones. Evaluation results demonstrate that TeamPhone can properly fulfill the communication requirements and greatly facilitate rescue operations.

The main contribution of this paper is the design, implementation and evaluation of TeamPhone. This contribution breaks down into the following aspects:

- We design TeamPhone which consists of a messaging system and a self-rescue system. The messaging system can accomplish different kinds of message transmissions with affordable delay and energy consumption, and the self-rescue system can send out emergency messages through energy-efficient grouping and wake-up scheduling.
- We implement TeamPhone as an app on the Android platform, and further validate TeamPhone's performance through comprehensive experimental evaluations.
- The design, implementation and evaluation are based on off-the-shelf smartphones, which enables TeamPhone to be installed as a factory default application on smartphones by manufacturers to facilitate rescue operations in disaster scenarios.

The rest of this paper is structured as follows. We start with the motivation and challenges, and then present the design and implementation, followed by experimental evaluations. After that, we discuss the on-going work, review the related work and conclude the paper.

II. Motivation and Challenges

In this section, we first motivate the need of networking smartphones in disaster recovery for communications, and then illustrate the challenges.

A. Motivation

Disasters, such as earthquakes, may topple countless homes and kill thousands of people. Power failures and fallen cellular towers caused by disasters further leave the affected area cut off from the outside and hinder rescue operations. In disaster recovery, communications are crucial for coordinating rescue operations. Moreover, if trapped survivors in the rubble can send out emergency messages to rescue workers, rescue operations can be greatly accelerated. Therefore, in this paper, we investigate how to provide communications in disaster recovery.

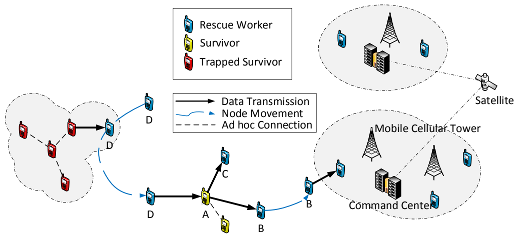


Fig. 1: Network scenario in disaster recovery

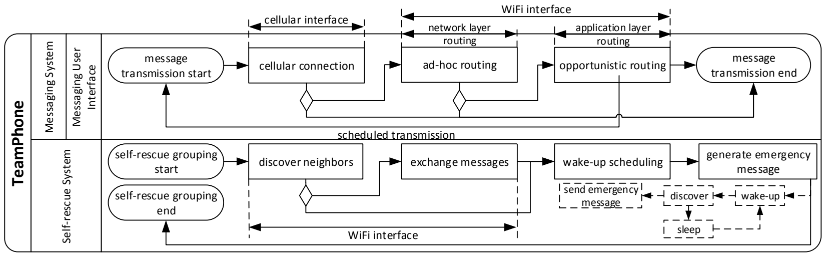


Fig. 2: Overview of TeamPhone

With the increasing penetration of smartphones equipped with short-range radios, GPS and sensors, smartphones have been used for various applications, including mobile sensing [15][20], ad-hoc communications [5][18], WiFi-based localization [10][21], *etc.*. Moreover, smartphones have evolved to be much more powerful than before in terms of computing and communications and users heavily rely on smartphones in their daily lives. As a result, users always carry their smartphones or place smartphones where they can easily and immediately be accessed even during disaster recovery. However, in disaster recovery such as earthquakes, the cellular towers may be destroyed, and thus cellular communication of smartphones is cut off. Then, we have to set up communication with short-range radios (*e.g.*, WiFi) in most smartphones.

Smartphones have recently been conceptually considered for disaster recovery to locate immobilized survivors using Bluetooth in [19] and to provide multi-hop communications in [14]. However, Bluetooth has limited communication range (a few meters). The design in [19] fails to consider energy efficiency, which may quickly drain the battery of the smartphone. [14] also fails to conserve energy and uses proactive routing which reacts slowly to the frequently changing network topology in disaster recovery. It also increases the maintenance overhead in terms of network traffic and energy consumption. Different from these existing works, we propose a more functional and energy-efficient communication system using smartphones for disaster recovery, and we address various design and implementation issues.

B. Challenges

During disaster recovery, communications satellites and mobile cellular towers may be deployed. However, communications satellites are scarce. Although mobile cellular towers can be used to set up the command center and provide critical communications for rescue workers, such vehicle carried towers cannot cover all disaster areas. Therefore, it is important to network smartphones with short range radios in disaster recovery.

Due to the mobility of rescue crews and survivors, network topology changes frequently; *i.e.*, sometimes smartphones may form a mobile ad-hoc network, and sometimes they only contact each other opportunistically. Therefore, the big challenge is how to provide communication crossing different types of networks including ad-hoc network, opportunistic network, and cellular network, considering frequent topology change; *i.e.*, rescue workers can communicate with each other and with the command center no matter they are within the coverage of the mobile cellular towers or not.

In addition, trapped survivors may be buried in the debris and be difficult to discover. With the capability of communications between smartphones, trapped survivors can send out emergency messages to nearby rescue crews with better discoverability and reachability. Moreover, they can also send out location information to better facilitate rescue operations. However, broadcasting emergency messages can drain batteries quickly. Since rescue operations may last for days after disasters occur, how to discover nearby rescue crews and send out emergency messages in an energy-efficient manner is another challenge.

III. TeamPhone

TeamPhone is a tailored system for disaster recovery based on smartphones, which provides seamless data communication through several different types of networks and facilitates rescue operations for trapped survivors. In this section, we describe the network scenario of disaster recovery, and present the design of TeamPhone.

A. Network Scenario

Fig. 1 depicts the network scenario for disaster recovery. As can be seen, mobile cellular towers only cover limited areas and provide cellular communications for rescue workers in that area. A command center sits in the covered region and command centers in different regions may be connected via communications satellites. Survivors can also join the network to help data communications. Trapped survivors in the debris are unable to move and are waiting for rescue.

When rescue workers fall out of cellular coverage, they can only use short-range radios (*e.g.*, WiFi) to communicate. They can communicate with each other or with the command center via individuals or combinations of ad-hoc connections, opportunistic contacts, and cellular connections, as shown in Fig. 1. Also, trapped survivors can construct a group based on ad-hoc connections and send out emergency messages when rescue workers or survivors are within the communication range of the group. Therefore, the exploitation of smartphones can greatly extend the communication field far beyond the region covered by mobile cellular towers, and increase the opportunity of being discovered and rescued for trapped survivors.

B. Overview

As shown in Fig. 2, TeamPhone includes two components: the messaging system and the self-rescue system. The messaging system runs on the smartphones of rescue workers or survivors, which provides message transmissions. The self-rescue system runs on the smartphones of trapped survivors, which automatically form groups with nearby trapped survivors and sends out emergency messages.

TeamPhone works as follows. First, users need to specify which system to use. If smartphones are specified as part of the messaging system (we call them messaging nodes), users can send messages via their smartphones; their smartphones will act as relays for both ad-hoc routing and opportunistic routing, and as gateways when they have cellular connections. Trapped survivors configure their smartphones as part of the self-rescue system (we call them self-rescue nodes), which can be triggered by other apps that measure seismic wave, such as *iShake* [8]. Self-rescue nodes are grouped and scheduled to wake up and receive *hello* messages from messaging nodes. When self-rescue nodes receive *hello* messages, they will reply with an emergency message and then the messaging node can send the emergency message to the command center.

C. Messaging System

The messaging system is designed to handle message transmissions via three ways: through cellular connections, by ad-hoc communications and upon opportunistic contacts. When a messaging node needs to transmit a message (text, voice, photo, *etc.*), it first tries to reach the destination via the cellular network. The message can be delivered only when both the source and destination are within the region covered by mobile cellular towers. If this fails, the messaging system will try to reach the destination by both the ad-hoc network and cellular network, *i.e.*, the messaging node will issue a routing request to construct a routing path to the destination based ad-hoc communications and cellular connections. If the request is fulfilled, the message can be directly sent to the destination.

Since messaging nodes that have cellular connections act as gateways in the ad-hoc network, they can be exploited to reach the destinations that have cellular connections, such as the command center. Therefore, the message might be transmitted to the gateway by ad-hoc connections and then relayed by the gateway to the destination by cellular connections. If there is no routing path to the destination but the command center can be connected, the message will be sent to and stored at the command center and it will be delivered when the destination contacts the command center. Otherwise, the message will be stored locally and transmitted upon opportunistic contacts between messaging nodes, and different opportunistic routing strategies can be applied.

Ad-hoc Routing. To reach the command center via ad-hoc connections, messages must be relayed at a gateway, *i.e.*, a messaging node in the cellular region. Suppose a reactive routing protocol is employed (*e.g.*, AODV routing). The messaging nodes will reply to the routing request differently, based on whether they are within or outside of the cellular region. Due to node movement, messaging nodes need to monitor the status of their cellular connection and configure themselves as gateways for the routing protocol when they have cellular connections, or vice versa.

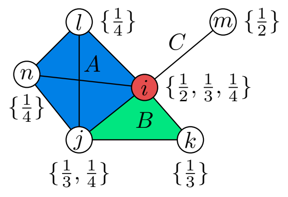
In the following, we use AODV as an example to illustrate how to bridge the ad-hoc network and cellular network via ad-hoc routing in our design. When a gateway receives a routing request for a destination, if it does not have an active path to the destination, besides forwarding the routing request to its neighbors, it will send back a routing reply with a 'gateway' flag and the hop count to the source. If the source receives a routing reply with an ad-hoc path to the destination, it will ignore the routing replies with the 'gateway' flag. Otherwise, the source will select the gateway with the minimum hop count, encapsulate the message and send it to the gateway. When the gateway receives the message, it will decapsulate it and try to connect the destination through its cellular interface.

Opportunistic Routing. In the messaging system, opportunistic routing acts as an alternative when the destination cannot be connected via cellular or ad-hoc communications. Opportunistic routing works as an application that forwards messages between two nodes that encounter each other. The opportunistic routing of the messaging system adopts two simple forwarding strategies: *(i)* static routing where the message carried by a messaging node is forwarded only when it encounters the destination, to save network resources such as energy and bandwidth; *(ii)* flood routing where messaging nodes that carry the message always forward it to the encountered node such that the delay of the message can be minimized.

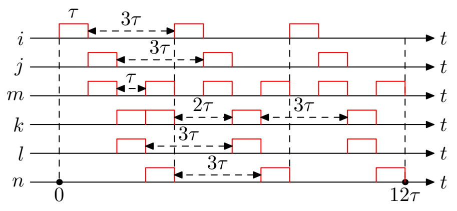
Unlike mobile opportunistic networks where nodes are isolated individually, the network in disaster recovery most likely consists of groups of nodes (*e.g.*, groups of rescue workers), where nodes within the group are well connected via ad-hoc communications and nodes between groups can also be connected through other nodes, *e.g.*, smartphones of survivors. In such a scenario, ad-hoc routing is preferred for message transmission rather than opportunistic routing. Therefore, in the messaging system, opportunistic routing is explored only when network partitions occur and thus two simple routing strategies are adopted rather than other sophisticated schemes based on historical contact information, such as [11][12].

D. Self-rescue System

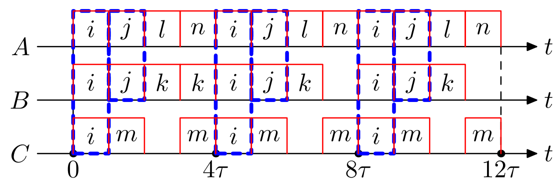
Trapped survivors configure their smartphones as part of the self-rescue system and then the self-rescue system automatically sends out emergency messages when rescue workers or survivors are nearby. The battery life of smartphones must last as long as possible, since rescue operations may last for hours or even days. Therefore, the self-rescue system must be energy-efficient. Since trapped survivors are most likely difficult to discover, rescue crews may not infer the location of trapped survivors, even if they have received emergency messages from them. Thus, the emergency message should also provide location information to facilitate rescue operations.



(a) one-hop network of i

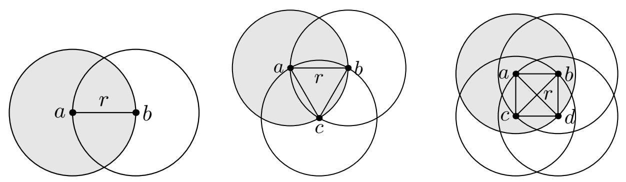


(b) wakeup schedule for each node

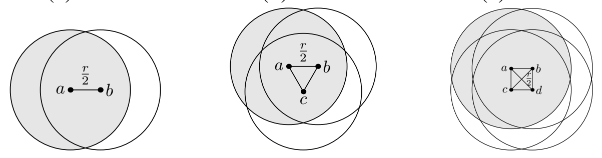


(c) cumulative wakeup schedule for each clique

Fig. 3: Illustration of one-hop network, individual wakeup schedule and cumulative wakeup schedule of cliques.



(a) 62.2% (b) 48.8% (c) 48.9%



(d) two-node clique (e) three-node clique (f) four-node clique

Fig. 4: Illustration of wireless coverage ratios if nodes wake up alternatively in two-node clique, three-node clique and four-node clique, where r is the transmission radius.

Self-rescue Grouping and Wake-up Scheduing. To save energy, instead of continuously staying awake, a self-rescue node can wake up periodically to discover messaging nodes. However, this will increase the possibility that a self-rescue node is asleep when a messaging node passes by. Since survivors in the same building may be trapped together or nearby when the building collapses, they can group together as a *self-rescue group* via WiFi and wake up in a coordinate way to save energy.

The group coordination must be carefully done, since they may still miss passing by nodes if only one node in the group is scheduled to wake up during some time. For example, as shown in Fig. 1, node D moves around the group of trapped survivors. If the leftmost node is scheduled to wake up and all others are sleeping, the self-rescue node will not receive the *hello* message since it is out of communication range of node D. Therefore, the design of the self-rescue system should take this into consideration. In the following we present the details of our solution.

Self-rescue nodes broadcast beacon messages to know their neighbors. Let N_u denote the set of one-hop neighbors of node u. Then, they broadcast their one-hop neighbor set such that all nodes know their two-hop neighbors. Note that self-rescue nodes do not consider unidirectional links; *i.e.*, for example, if node u receives the one-hop neighbor set of node v but N_v does not include u, node u will not count node v as a one-hop neighbor. Based on the information of two-hop neighbors, they can construct one-hop networks that include one-hop neighbors and edges among them, for example, as shown in Fig. 3a, which is built by node i.

One-hop networks consist of cliques, for example, in Fig. 3a, there are three cliques in the network, which are $A = \{i, j, l, n\}$, $B = \{i, j, k\}$ and $C = \{i, m\}$. In a clique there exits an edge between every two nodes, and hence any node in a clique can cover all other nodes. Therefore, in a clique nodes are close to each other, and the area covered by one node can be a large proportion of the area covered by all nodes in the clique.

Fig. 4 illustrates the ratios between the coverage of one node and the coverage of all nodes in two-node, three-node and four-node cliques. In Figures 4a, 4b and 4c, the percentage indicates the least coverage ratio for each case; *i.e.* the least coverage ratio for two-node clique, three-node clique and four-node clique are 62.2%, 48.8% and 48.9%, respectively, when the longest distance between nodes is exactly the transmission radius r. The coverage ratio increases with the decrease of such distance. For example, when it is decreased to $r/2$, as shown in Figures 4d, 4e and 4f, the coverage ratio of one node greatly increases and it is close to the coverage of all nodes in the clique. Thus, instead of waking up individually, nodes in a clique should wake up alternately (*i.e.*, there is no more than one awake node at any time in a clique) to discover nearby messaging nodes so as to save energy.

A node may belong to multiple cliques and thus the node should determine a wake-up schedule across all these cliques. For example, in Fig. 3a, node i belongs to cliques A, B and C and the wake-up fraction of node i is $1/4$ (since there are four nodes in clique A), $1/3$ and $1/2$ for A, B and C, respectively. To save energy, for each node, we choose the lowest wake-up fraction of the cliques it belongs to, denoted as γ, as the wake-up frequency (*e.g.*, $\gamma_i = 1/4$ for node i) and the sum of wake-up fractions of the cliques as the criterion, denoted as θ, to determine the scheduling order of nodes since the wake-up schedules of nodes in the same clique are dependent. In a clique, the wake-up schedule of the node with large θ is determined first.

Fig. 3b shows the wake-up schedule of each node in the network of Fig. 3a, assuming the wake-up schedule starts at time 0 and each wake-up time period is τ, which is a system parameter. The wake-up schedule of the nodes is easily determined following the scheduling order discussed above and two rules: *(i)* the wake-up frequency for each node must be γ; *(ii)* there is no more than one wake-up node at any time in

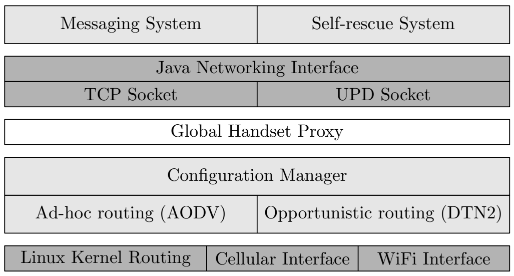
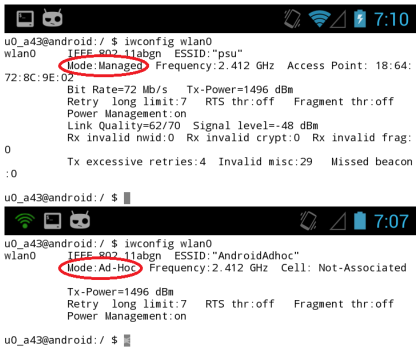


Fig. 5: Architecture of TeamPhone Implementation

a clique. Fig. 3c shows the wake-up schedule for each clique, which is the cumulative schedule of nodes in each clique. We can see for cliques B and C, there are vacancies because node i is scheduled at $\gamma_i = 1/4$ rather than $1/2$ in clique C and node j is scheduled at $\gamma_i = 1/4$ rather than $1/3$ in clique B. Since the vacancy ratio is low, *i.e.*, 16.7% for clique B and 25% for clique C, and the coverage ratios are high as discussed above, the schedule vacancy only slightly affects the overall coverage; yet such wake-up scheduling can greatly save energy. Nodes may belong to more than one clique, such as nodes i and j in Fig. 3c. When such nodes are awake, no other nodes in its clique need to wake up to save energy.

In the following, we describe how to determine the wake-up schedule in a distributed way. As discussed above, by exchanging neighboring information, each node can construct the one-hop network, and it can further calculate θ and γ based on the cliques it belongs to. Then each node floods θ into the network, and after that each node acknowledges θ of all other nodes. The node that has the maximum θ will initiate the scheduling procedure; *i.e.*, it decides a reasonable start time for the wake-up schedule, determines its own wake-up schedule based on γ, and then broadcasts the schedule to its one-hop neighbors. For each node (except the initiator), it determines and then broadcasts its own schedule only when it has received all the schedules from the nodes with larger θ in the same clique. Since the initiator of the scheduling procedure decides the start time of the wake-up schedule, clock synchronization may be required for all the nodes in the self-rescue group. However, smartphones usually get the local time from network providers such that self-rescue nodes are already synchronized with millisecond accuracy before disasters occur. Therefore, no additional synchronization is required since τ is at second level.

Emergency Message. Each node acknowledges the number of nodes in the group based on the received information. To better facilitate rescue operations, emergency messages should also include location information. Therefore, during wake-up scheduling, each node also includes its current location if a GPS signal is available, or last known location from either GPS or the network provider, into the flood of θ. Since many Apps provide location-based services with access to user location running in the background, the last known location of self-rescue nodes may be obtained just before the disaster occurs, and thus it can be useful to infer the current location of self-rescue nodes. In summary, an emergency message


```
u0_a43@android:/ $ iwconfig wlan0
wlan0     IEEE 802.11abgn  ESSID:"psu"
          Mode:Managed  Frequency:2.412 GHz  Access Point: 18:64:
72:8C:9E:02
          Bit Rate=72 Mb/s   Tx-Power=1496 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=62/70  Signal level=-48 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:
0
          Tx excessive retries:4  Invalid misc:29   Missed beacon
:0

u0_a43@android:/ $
```

```
u0_a43@android:/ $ iwconfig wlan0
wlan0     IEEE 802.11abgn  ESSID:"AndroidAdhoc"
          Mode:Ad-Hoc  Frequency:2.412 GHz  Cell: Not-Associated

          Tx-Power=1496 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on

u0_a43@android:/ $
```

Fig. 6: WiFi status when being configure as managed mode and ad-hoc mode

includes: *(i)* the number of trapped survivors; *(ii)* when have they been trapped; *(iii)* the latest known location of each node.

IV. TeamPhone Implementation

In this section, we describe the detailed implementation, which includes TeamPhone interface, TeamPhone routing, and TeamPhone application, where the network interface configuration and routing are implemented in C, C++ based on Linux, and the application is implemented in Java based on Android. The architecture of TeamPhone implementation is illustrated in Fig. 5.

A. The Interface

Besides the cellular interface, smartphones are usually equipped with Bluetooth and WiFi. Since Bluetooth has limited transmission range, WiFi is used for ad-hoc communications between smartphones. However, WiFi ad-hoc mode is not officially supported by Android. To enable WiFi ad-hoc mode, we need to compile wireless extension support into the Linux kernel and also compile the wireless tool *iwconfig* for smartphones, which will be utilized to configure the WiFi driver. The configuration of WiFi includes switching WiFi from managed mode (also known as station infrastructure mode), which is the default mode of WiFi on smartphones and cannot be used for ad-hoc communications, to ad-hoc mode, and manipulating sleep and wake-up periods of WiFi. Fig. 6 illustrates the status of WiFi on smartphones when being configured as managed mode and ad-hoc mode. In TeamPhone, ad-hoc routing exploits both the cellular interface (in gateway mode) and WiFi interface, while opportunistic routing only employs the WiFi interface.

B. TeamPhone Routing

AODV. In TeamPhone, we modified the Linux implementation of AODV [1] and cross-compiled it for smartphones. The implementation includes two components: a loadable kernel module and a user-space daemon. The kernel module captures data packets using *Netfilter* with POSTROUTING, which requires the Linux kernel with *Netfilter* enabled, and instructs the user-space daemon to issue a routing request if the destination of the packet is out of route entries. In addition, the kernel module is in charge of encapsulating and

Parameter	Setting
hello broadcast interval	1s
# of *hello*s before treating as neighbor	3
Allowed *hello* loss	2
Routing request timeout	4s
Routing request retries	2
Routing reply ACK timeout	50ms
Active route timeout	10s

TABLE I: Parameter settings of AODV

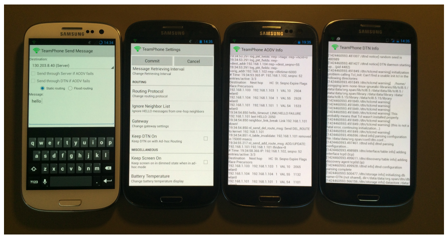


Fig. 7: Testbed

decapsulating the packet that goes through gateways. The user-space daemon maintains the kernel routing table and controls neighbor discovery, routing request and routing reply. The communications between these two components are based on *NetLink* socket.

In disaster recovery, messaging nodes monitor the availability of cellular connections and configure themselves as gateways for AODV if the cellular connection is valid. To maximize the reachability of messages, if the destination cannot be reached by AODV, the message can be temporarily stored at the command center through gateways and the message will be delivered to the destination once it connects to the command center. We have integrated this function into the implementation of AODV. Table I shows the settings of major parameters of AODV in our implementation.

DTN2. In TeamPhone, opportunistic routing is accomplished by DTN2 [2], which is a Linux implementation of the DTN bundle protocol defined in RFC 5050 [4]. We customized and cross-compiled DTN2 for smartphones. DTN2 is a routing protocol working at the application layer and running as a user-space daemon. The network interface configured for DTN2 is WiFi in ad-hoc mode. The basic workflow of DTN2 on smartphones is as follows. First, a convergence layer needs to be configured, which is used to transmit messages between smartphones. We choose the TCP convergence layer to guarantee the message transmission. Next, service discovery will advertise the convergence layer's presence to neighbors. Meanwhile a node also listens for neighbor beacons and distributes each event of neighbor discovery to the convergence layer. Then, DTN2 performs message transmissions with neighbors according to the configured routing protocol. Static routing and flood routing are two currently supported routing protocols in TeamPhone.

C. TeamPhone Application

The TeamPhone application wraps the messaging system and self-rescue system together, implemented in Java on Android system. When TeamPhone is launched, WiFi will be configured as ad-hoc mode and users need to specify which system to use.

For the messaging system, AODV and DTN2 are initialized for routing, and the messaging app is provided for users to send and receive messages. Messages can be sent in three ways: *(i)* through AODV to reach the destination; *(ii)* by gateways to store the message at the command center from which the message will be eventually delivered when the destination connects to the command center (*e.g.*, nodes periodically fetch messages from the command center directly or through gateways); *(iii)* by DTN2 in static routing mode and flood routing mode. In the prototype, the first method is the default and other two are optional. The messaging system is also designed to receive emergency messages from self-rescue nodes. The *hello* message from the messaging node and self-rescue node is flagged differently so as to deal with the unidirectional link between messaging node and self-rescue node, where only the messaging node can receive *hello* messages from the self-rescue node. In that case, the messaging node will not receive the emergency message from the self-rescue nodes since the self-rescue nodes are unable to discover the messaging node. The messaging node can still be alerted by receiving the *hello* message, and this alert function is implemented together with the user-space daemon of AODV.

The self-rescue system performs self-rescue grouping and wake-up scheduling in the background. For the case that trapped survivors may not have the opportunity to start self-rescue system manually, TeamPhone can employ *iShake* [8], which exploits smartphones as the seismic sensor, to trigger the self-rescue system automatically when an earthquake occurs. After self-rescue grouping and wake-up scheduling, self-rescue nodes can configure the determined wake-up schedule of WiFi using *iwconfig*; *i.e.*, they need to enable WiFi power management and set up the period between wake-ups and the timeout before going back to sleep (*i.e.*, λ). When self-rescue nodes are awake, they run AODV to detect messaging nodes in vicinity and send out the emergency message once they discover the messaging node. The function of sending emergency messages is embedded into the daemon of AODV.

V. Evaluations

TeamPhone is deployed on the off-the-shelf Android smartphones, *i.e.*, Samsung Galaxy S3. We build a small testbed, as shown Fig. 7, to evaluate the performance of TeamPhone. We do note that there exist designs for communications in disaster recovery. However, few of them are implemented as real systems and none provides system evaluation. Thus, we cannot compare them with TeamPhone in the evaluation. As discussed in Section II-A, TeamPhone obviously outperforms these works in many aspects of design.

A. Messaging System

In the experiments, one smartphone has a cellular connection and works as a getaway. We also deploy a server

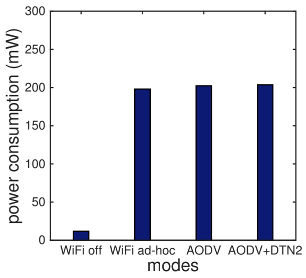


Fig. 8: Power consumption when the smartphone is in different modes

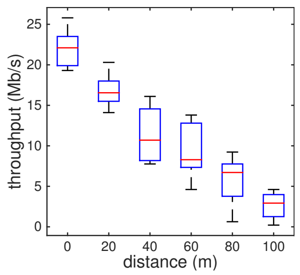


Fig. 9: Throughput as a function of distance between two directly connected smartphones

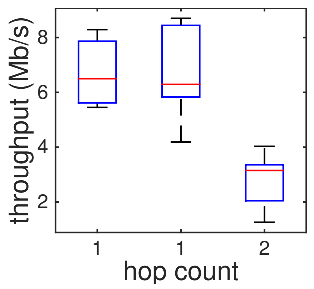


Fig. 10: Throughput vs. hop count

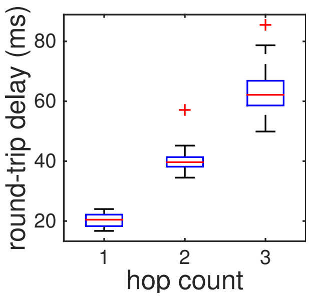


Fig. 11: Round-trip delay of AODV on smartphones

which can be connected through the cellular network by smartphones to store and forward messages. This server acts as the command center in disaster recovery. Fig. 7 illustrates the user interface for sending messages, settings, and the log information of AODV and DTN2. In the messaging system, messages are transmitted over TCP connections (TCP convergence layer for DTN2) to guarantee the delivery. The evaluation demonstrates that TeamPhone can accomplish the following ways of message transmissions:

- Sending by AODV (including through gateways)
- Sending by AODV-Gateway-Server-Gateway-AODV
- Sending by DTN2

In the following, we evaluate the messaging system in terms of power consumption, throughput and delay, which are important aspects of a communication system in disaster recovery. We also show how opportunistic routing can facilitate messaging transmissions and obtain better performance than with only ad-hoc routing.

Power Consumption The message system should be energy-efficient. Fig. 8 illustrates the power when the smartphone is operating in different modes, where all the measurements are conducted when the smartphone's screen is off and the power is averaged over one minute. As can be seen, when WiFi is off, the smartphone consumes about 10mW. When WiFi is on and configured in ad-hoc mode, the power consumption increases significantly, to about 200mW, because WiFi in ad-hoc mode has to be at the working stage to transmit and receive messages. When the messaging system with AODV (the *hello* message interval is one second) is running on the smartphone without user-space data traffic, it incurs very little additional power. Similarly, the running of DTN2 only consumes very little power. From Fig. 8, we can conclude that the basic power consumption of the messaging system (with both AODV and DTN2) is about 200 mW, which is mainly consumed by the WiFi module. For the Samsung Galaxy S3 with battery of 2100mAh and 3.8V, the estimated standby time is about 40 hours when running the messaging system.

The *hello* message interval of AODV is set to one second based on the following considerations. When AODV sends *hello* messages more frequently, such as two *hello* messages per second, it will cause more network traffic, especially when the network has many nodes. When *hello* messages are sent less frequently, AODV will react to the change of network topology slowly and thus in turn affect the performance of AODV. For example, when the *hello* interval is four seconds, the reaction delay to a neighbor change is about 8 seconds, since the allowed loss of *hello* messages is 2 as in Table I. Moreover, as observed from the experiments, the power consumption for different *hello* message intervals only varies slightly. Therefore, in TeamPhone, the *hello* message interval is set to one second.

Throughput. Since transmitted messages could be photo, voice, even video, we also measure the throughput of the messaging system as shown in Figures 9 and 10, where the WiFi module of the smartphone S3 supports IEEE 802.11 a/b/g/n. Fig. 9 illustrates the throughput between two directly connected smartphones. The maximum throughout is over 20Mb/s, and it decreases linearly with the increase of the distance between two smartphones. When the transmission distance is 100 meters, the throughput drops to about 2Mb/s. Fig. 10 shows the throughput on a two-hop ad-hoc path, constructed by AODV. As can be seen, the throughput of both one-hop paths is about 6Mb/s. The throughput of two-hop drops to 3Mb/s due to the increase of hop count.

Delay. Fig. 11 shows the round-trip delay of message passing with AODV. As expected, the delay increases with the hop count, and the delay for one-hop message passing is low (about 20ms). For two-hop and three-hop, the delay is about 40ms and 60ms, respectively. Unlike one-hop, where neighbors are already in the routing table, multi-hop requires routing path discovery if the destination is not in routing table. Thus, the delay variations of two-hop and three-hop are much higher than the median values as shown Fig. 11.

Benefits of Opportunistic Routing. The incorporation of opportunistic routing is designed to enhance the performance of the messaging system as demonstrated in Fig. 12 and 13.

In Fig. 12, node S needs to send a message to node D. Node S first performs path discovery of AODV. However, none of S, R_1 and R_2 can contact D due to a physical obstacle as shown in Fig. 12. Therefore, the connection between S and D cannot be established using AODV (or any ad-hoc routing) and the message transmission is failed. Such a scenario is common in disaster recovery. Nevertheless, if opportunistic routing is deployed, the message can be replicated at R_1 and R_2 first since R_1 and R_2 can be reached from S, and then the

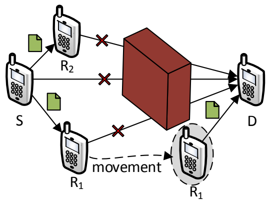


Fig. 12: Movement helps data transmission

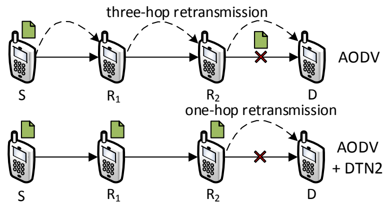


Fig. 13: Replication reduces transmission delay

message can be delivered when any of them moves closer to D and establishes a connection with D. Therefore, opportunistic routing can take advantage of node movement to accomplish message transmissions, and the delay is mainly determined by the physical movement.

In Fig. 13, node S is sending a message to D through multi-hop connection between them. However, the link between R_2 and D is suddenly broken due to channel interference, or physical obstacles. If only ad-hoc routing is used. S has to re-transmit the message to D when the link between R_2 and D is back, and a new route discovery has to be initiated. With opportunistic routing, the messaging system can handle such a scenario with better performance as follows. When the link between R_2 and D is broken, the messaging system can distribute the message to other nodes (*i.e.*, R_1 and R_2) on the path to the destination. When the link is valid again, the message can be directly transmitted to node D by R_2 instead of three-hop communications from S to D. Therefore, message replication of opportunistic routing can greatly accelerate message transmissions. Based on the experiments on the testbed, for the scenario in Fig. 13, the round-trip delay can be decreased from about 60ms to 20ms, and the throughput can be increased from about 1Mb/s to more than 6Mb/s.

From these two scenarios, we can see opportunistic routing can greatly facilitate message transmissions and improve the performance of the messaging system.

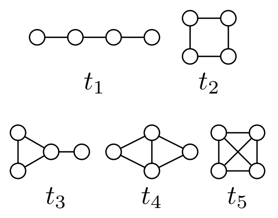
B. Self-rescue System

For the self-rescue system, we first need to determine the wake-up period τ. As self-rescue nodes wake up to discover the messaging node, τ should be long enough such that they can receive the *hello* message from the messaging node and send out the emergency message. Since self-rescue nodes wake up alternately in a clique, τ should be short enough so that the self-rescue node is awake when messaging nodes are passing by. Based on these considerations and the parameter settings of AODV, τ is set to 5 seconds in the experiments.

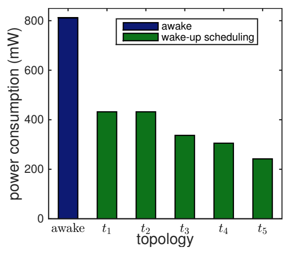
	Power (mW)	**Energy (mJ)**	**Time period (s)**
Wake-up	202.30	1011.5	5
Sleep	12.98	64.9	5

TABLE II: Power and energy consumption of the self-rescue system

Table II shows the power and energy consumption of the self-rescue system. When nodes are asleep (WiFi goes to sleep), they consume very little power, similar to that when WiFi is turned off. When they are awake, the power consumption is similar to that of messaging nodes as in Fig. 8. According to the wake-up scheduling, the worst case of energy consumption is when the sleep period of a self-rescue node is also τ (*i.e.*, the self-rescue node only belongs to a two-node clique). In that case, the energy consumption is about 1076mJ for a sleep period and a wake-up period (totally 10 seconds) as shown in Table II.



(a) network topology of self-rescue group



(b) power consumption of different topologies

Fig. 14: Power consumption of self-rescue group

To measure the power consumption of the self-rescue group, smartphones form groups with different network topologies, as shown in Figure 14a. There are five different network topologies, denoted as t_1, t_2, t_3, t_4 and t_5. For each topology, we first determine the wake-up schedule and then measure the total power consumption of the group. In Fig. 14b, *awake* indicates the power consumption when all the nodes stay awake. As can be seen in Fig 14b, our wake-up scheduling is always better than *awake*, and the power consumption is less than half of *awake*. When the network contains a larger clique, the self-rescue group consumes less power. For example, as t_3 contains a three-node clique while t_2 contains a two-node clique, the power consumption of t_3 is less than that of t_2. Similarly, t_5 (four-node clique) is less than t_4 (three-node clique). Moreover, the power consumption of t_5 is only $1/3$ of *awake*. For smartphone S3, the standby time of the group when running the self-rescue system is about 70 hours for t_1 and 125 hours for t_5. Therefore, we can conclude that the self-rescue system can greatly save energy and thus increase the possibility of being discovered in rescue operations for trapped survivors.

VI. On-going Work

TeamPhone enables smartphone communication in disaster recovery. It can be installed as a factory default application and can greatly help survivors and rescuers. Although several ways of message transmissions have been provided, there may not be a server in the network and thus we are currently exploring the design of a routing scheme that can directly concatenate gateways in different network partitions. Specifically, we are working on AODV-gateway-gateway-AODV routing. That is, when a gateway receives a routing request, it not only relays the data locally but also sends it remotely to other gateways, and then other gateways can circulate the routing request in their local networks. Eventually, routing paths that cross different network interfaces and different network partitions can be established.

The self-rescue system currently includes the last known location of self-rescue nodes into the emergency message. However, such location information might not be accurate. We are currently investigating how to exploit WiFi signals, especially Channel State Information, to accurately position self-rescue nodes. Then, when messaging nodes receive the emergency message, the location of each self-rescue node can be inferred based on the position information of self-rescue nodes and the location of messaging nodes that can be easily obtained by GPS. Such localization can greatly facilitate rescue operations.

VII. Related Work

Research efforts have been made to apply communication technologies to disaster recovery. Many researches focus on using ad-hoc networks. Zussman *et al.* [22] proposed to employ the network formed by smart badges to collect information from trapped survivors. Reina *et al.* [17] extensively evaluated ad-hoc routing protocols in disaster scenarios. Aschenbruck *et al.* [6] modeled mobility in disaster areas and investigated how the model affected network performance of ad-hoc routing protocols. Other researches take opportunistic networks into consideration. Martín-Campillo *et al.* [13] proposed a random walk gossip protocol that runs over ad-hoc networks. Asplund *et al.* [7] evaluated the efficiency of opportunistic routing protocols in disaster scenarios. Fujihara and Miwa [9] proposed disaster evacuation guidance using opportunistic communications. A more detailed survey of ad-hoc neworks for disater scenarios can be found [16].

Although many researchers have contributed to improving rescue and evacuation operations, only few consider smartphones. In the literature, [19] and [14] are the most related work to this paper. Suzuki *et al.* [19] proposed a design to assist the search for immobilized persons in a disaster area using Bluetooth of smartphones. However, Bluetooth of smartphones only has limited communication range (maximum 10m), and the design does not consider energy efficiency and can quickly drain the battery.

Nishiyama *et al.* [14] assume that smartphones can be charged by mobile solar cells and thus did not consider energy efficiency. However, in disaster scenarios, solar cells may not be accessible for survivors, and thus the designed system can only provide temporary communications. Among the related works, few are implemented as real systems and none provides system evaluation. However, TeamPhone is designed, implemented, and evaluated based on off-the-shelf smartphones and ready to be installed on smartphones to provide communications and facilitate rescue operation in disaster scenarios.

VIII. Conclusions

In this paper, we propose TeamPhone, which is designed to network smartphones in disaster recovery. TeamPhone includes two components: the messaging system that provides data communications for rescue workers, and the self-rescue system that groups the smarphones of trapped surviors together and energy-efficiently discovers nearby messaging nodes and sends out emergency messages. TeamPhone is implemented as a prototype application on the Android platform using the WiFi interface and cellular interface to provide several ways of communications. TeamPhone has been deployed and evaluated on the off-the-shelf smartphones. The evaluation results demonstrate that TeamPhone can accomplish various message transmissions with affordable power consumption and delay, and greatly reduce the energy consumption of sending out emergency messages by grouping and wake-up scheduling.

References

[1] AODV-UU. http://aodvuu.sourceforge.net.

[2] DTN2. http://www.dtnrg.org.

[3] Powerful quake ravages china, killing thousands. http://www.nytimes.com/2008/05/13/world/asia/13china.html.

[4] RFC 5050. https://tools.ietf.org/html/rfc5050.

[5] A. Al-Akkad, L. Ramirez, A. Boden, D. Randall, and A. Zimmermann. Help beacons: design and evaluation of an ad-hoc lightweight sos system for smartphones. In *Proc. CHI*, 2014.

[6] N. Aschenbruck, M. Frank, P. Martini, and J. Tolle. Human mobility in manet disaster area simulation-a realistic approach. In *Proc. LCN*, 2004.

[7] M. Asplund and S. Nadjm-Tehrani. A partition-tolerant manycast algorithm for disaster area networks. In *Proc. SRDS*, 2009.

[8] M. Ervasti, S. Dashti, J. Reilly, J. D. Bray, A. Bayen, and S. Glaser. iShake: Mobile phones as seismic sensors – user study findings. In *Proc. MUM*, 2011.

[9] A. Fujihara and H. Miwa. Disaster evacuation guidance using opportunistic communication: The potential for opportunity-based service. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 425–446. 2014.

[10] K. Liu, X. Liu, and X. Li. Guoguo: Enabling fine-grained indoor localization via smartphone. In *Proc. MobiSys*, 2013.

[11] Z. Lu, X. Sun, Y. Wen, and G. Cao. Skeleton construction in mobile social networks: Algorithms and applications. In *Proc. SECON*, 2014.

[12] Z. Lu, X. Sun, Y. Wen, G. Cao, and T. La Porta. Algorithms and applications for community detection in weighted networks. *Parallel and Distributed Systems, IEEE Transactions on*, 26(11):2916–2926, 2015.

[13] A. Martín-Campillo, J. Crowcroft, E. Yoneki, and R. Mart. Evaluating opportunistic networks in disaster scenarios. *Journal of Network and Computer Applications*, 36(2):870–880, 2013.

[14] H. Nishiyama, M. Ito, and N. Kato. Relay-by-smartphone: realizing multihop device-to-device communications. *Communications Magazine, IEEE*, 52(4):56–65, 2014.

[15] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan. Medusa: A programming framework for crowd-sensing applications. In *Proc. MobiSys*, 2012.

[16] D. Reina, J. Coca, M. Askalani, S. Toral, F. Barrero, E. Asimakopoulou, S. Sotiriadis, and N. Bessis. A survey on ad hoc networks for disaster scenarios. In *Proc. INCoS*, 2014.

[17] D. Reina, S. Toral, F. Barrero, N. Bessis, and E. Asimakopoulou. Evaluation of ad hoc networks in disaster scenarios. In *Proc. INCoS*, 2011.

[18] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura. Serendipity: enabling remote computing among intermittently connected mobile devices. In *Proc. MobiHoc*, 2012.

[19] N. Suzuki, J. Zamora, S. Kashihara, and S. Yamaguchi. Soscast: Location estimation of immobilized persons through sos message propagation. In *Proc. NICos*, 2012.

[20] Y. Wang, W. Hu, Y. Wu, and G. Cao. Smartphoto: a resource-aware crowdsourcing approach for image sensing with smartphones. In *Proc. MobiHoc*, 2014.

[21] Y. Wang, J. Liu, Y. Chen, M. Gruteser, J. Yang, and H. Liu. E-eyes: device-free location-oriented activity identification using fine-grained wifi signatures. In *Proc. MobiCom*, 2014.

[22] G. Zussman and A. Segall. Energy efficient routing in ad hoc disaster recovery networks. In *Proc. INFOCOM*, 2003.