Ad Hoc Networks 11 (2013) 2440–2460

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

DistressNet: A disaster response system providing constant availability cloud-like services ☆

Harsha Chenji [a], Wei Zhang [a], Radu Stoleru [a,*], Clint Arnett [b]

[a] Department of Computer Science and Engineering, Texas A&M University, United States
[b] TEEX Disaster Preparedness and Response, United States

ARTICLE INFO

Article history:
Received 21 July 2012
Received in revised form 21 June 2013
Accepted 25 June 2013
Available online 5 July 2013

Keywords:
Disaster response
Adhoc networks
Sensor networks
Delay tolerant networks
Cloud services

ABSTRACT

Large scale disasters like the earthquake and tsunami in Japan (2011) cripple the local infrastructure. Proprietary systems and protocols used today for disaster response still lack data at the high spatial and temporal resolution needed to quickly save lives and to support disaster recovery efforts. Victims are rescued after days, if not weeks; digital coordination interfaces among responders are lacking, or are based on archaic methods (pencil, paper, paint on walls); the delay in receiving vast amounts of information from the field is bounded by the time used to physically transport tapes or hard drives. In this paper we present the design, implementation and evaluation of DistressNet, a system that provides services for emergency response applications. DistressNet integrates a variety of rapidly deployable, battery powered COTS devices into a secure framework. An optimal placement of networked components allows users to quickly and reliably store and retrieve data, in a "cloud"-like manner, from a local intermittently connected "fog". High volumes of field data are available for emergency response personnel to view on interfaces like smartphones and tablets. DistressNet is a large academic effort, proposing open systems, instead of proprietary solutions. It has been developed in collaboration with Texas Task Force 1 and its components have been evaluated for over one year in outdoor deployments that required over 1500 man hours.

1. Introduction

Disasters, natural or man made, are unexpected events that cause significant distress and havoc on a global scale. The best that can be done in the face of such uncontrollable acts of nature is speedy and effective recovery. Recent disasters in Japan and Haiti [1,2] have shown the effect that they can have on people, property, and the economy. Repercussions include, but are not limited to shortage of electric power, food, potable water, protection from the elements of nuclear and/or chemical hazards. In such situations, disaster response becomes increasingly difficult and constrained.

Several countries have set up governmental agencies to deal with such disasters, such as Texas Task Force 1 – Urban Search & Rescue (US&R), an agency part of FEMA [3] in the US. Several Task Forces comprising of trained personnel and specialized equipment have been deployed by FEMA in the event of such disasters. From our collaborations with US&R responders, we are keenly aware of the 66 tons of equipment emergency responders maintain in their cache. While this equipment has been tried and tested in the field, there are numerous examples in which new technologies like deeply embedded sensors and ad hoc/delay tolerant networking over high capacity storage devices can make a significant impact. Some of these technologies have not been considered robust enough unless deployed by a military/government contractor until

☆ An earlier version of this article appeared in the 31st IEEE International Performance Computing and Communications Conference (IPCCC), 2012.

* Corresponding author.
E-mail address: stoleru@cse.tamu.edu (R. Stoleru).

1570-8705/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2013.06.008

recently, when the US Army announced that it plans to adopt commercially available hardware (e.g., iPhones) for combat [4].

Research challenges that need to be addressed before new such technologies are adopted in the real world, i.e., by emergency response teams, include secure transport and high availability of data to and from heterogeneous devices, the ability to synchronize data with cloud services like Amazon S3, Microsoft Azure or Flickr, modular and open design of infrastructure and applications, and a high-throughput delay tolerant network of devices powered by batteries. Additionally, emergency responders have specific requirements or applications. One such requirement, obtained in consultation with US&R responders is to enable the discovery of victims under the rubble of collapsed buildings in a timely manner (unlike several days in Japan).

To address the aforementioned challenges, we have designed, implemented and evaluated DistressNet, a wireless sensor, adhoc and secure delay tolerant network system for disaster response. Thousands of sensors, equipped with vibration and acoustic sensors, are deployed over all collapsed buildings, continuously monitoring them for potential survivors under the rubble. Buildings surveyed by US&R responders are digitally tagged, allowing for fast, reliable and inexpensive high resolution data collection and situational awareness. Data is stored and replicated locally on battery powered devices, as well as being simultaneously backed up to a traditional cloud service. Teams of responders are equipped with mobile computing devices which can readily access data from other areas in the disaster as well as collaborate with other teams. Data and generated events in the field are relayed over an open standard delay tolerant network to the Command and Control center (C2). Strategically placed data waypoints allow for a high throughput and secure data delivery paradigm. DistressNet runs entirely on batteries, as US&R emergency responders learned is necessary, during Hurricane Katrina. More precisely, the contributions of our paper are as follows:

- To the best of our knowledge, we present the first design and implementation of a complex system (i.e., sensing, networking, data management) for emergency response that addresses US&R responder requirements and is evaluated in a realistic environment, utilizing over 1500 man-hours worth of deployment experiences and data.
- A FogNet service which enables constant availability of external cloud services over a local, disconnected DTN.
- Development of sensing modality that allows continuous monitoring of a large number of collapsed buildings for survivors, in stark contrast with today's state of art, requiring responders to be physically present in the field, and requiring no noisy activity, interfering with their acoustic monitoring.
- A unified mathematical model that optimizes the aggregate throughput of data flows in a heterogeneous DTN based on vehicle mobility and per-contact data transfers, made possible by placing a few additional devices.

2. Motivation and related work

Our motivating scenario is a large scale disaster (e.g., entire cities/regions are affected) and not a local, block-wide emergency in a city or a town. Unfortunately, recent history gives a few motivating examples like the earthquake in Haiti [2] and the tsunami in Japan [1]. In these incidents, the communication infrastructure is disrupted (i.e., cellular networks are completely or partially damaged) for weeks if not months, there are serious shortages of power (i.e. power sources like nuclear reactors are damaged), surveying the disaster area for survivors under the rubble takes from days to weeks (with some inspiring examples of survivors emerging after tens of days) and the C2 is flooded with sensing and multimedia data from the field. This febrile, fast pace environment lasts from one to several weeks, until the infrastructure is usable again.

In this remaining part of this section, we briefly review related, state of art work. We have presented a more complete review in [5]. A mandated and standardized equipment list for FEMA US&R teams is available online [6]. Each task force maintains its own cache, containing over 16,000 items. The technical equipment details Project 25 (P25) [7] compatible 2 way portable wireless radios. A 120 V AC powered base station and battery powered repeaters are mentioned. Voice/data channels are available on such systems only at very low data rates of 9.6Kbps [8].

The Wireless Internet Information System for Medical Response in Disasters (WIISARD) [9–11] is a 802.11 based wireless mesh network (WMN) tailored to provide effective medical response in the event of a disaster. Digital tags on patients [12] are read by medical personnel using PDAs which roam the area while being connected to the Internet via backhaul connections [13]. The primary difference between DistressNet and the latest version of WIISARD [9] is that DistressNet is designed for the needs of urban search and rescue personnel when operations occur over a large geographical area. While medical triage is handled by WIISARD, DistressNet addresses US&R operations like searching for survivors building rubble and building monitoring using low power 802.15.4 devices. As stated in [9], the major contribution is not WCP itself, but the characterization of link quality and human mobility patterns during the medical triage phase. In [14], the authors present a framework designed to improve information sharing during disaster rescue. Responders place RFID tags on buildings and program them with 802.11 enabled devices, which form a mobile ad hoc network. Our work utilizes 802.15.4 based motes which are capable of sensing temperature and/or air quality – a feature that is not found in RFID tags.

We draw upon a large body of research experiences in the field of delay tolerant networking (DTN) [15]. Dieselnet and the DOME testbed [16] provide rich information about implementing routing protocols [17] and providing a public DTN testbed using WiFi devices mounted on buses. Project RESCUE [18] provides an overview of a WMN [19,20] for effective emergency response. In [21], a hybrid

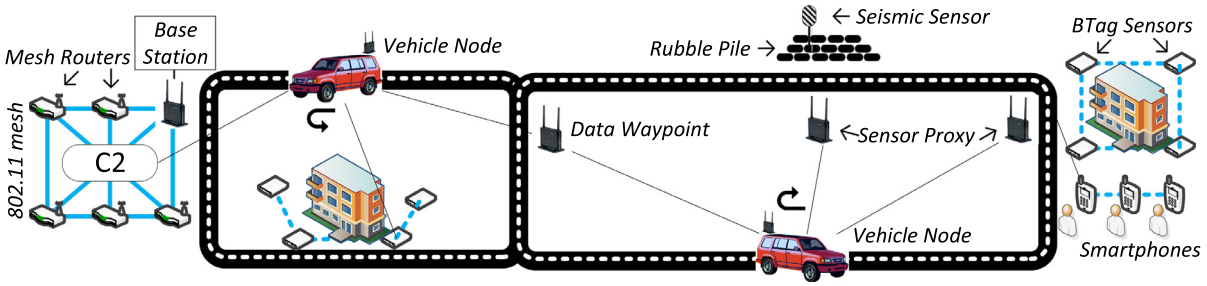


Fig. 1. Schematic of a DistressNet deployment showing all components. Data generated by BTag and Seismic Sensors is ferried to the Base Station using Vehicle Nodes. A Data Waypoint improves the data transfer process by creating a contact opportunity between two Vehicle Nodes.

WMN makes use of wireless WANs to access traditional networks using routers affixed to lamp posts. The SAFIRE project [22] deals with situational awareness for firefighters. [23] has commercial offerings which accomplish network centric warfare. To the best of our knowledge, these systems assume a powered, connected network and do not offer integration of low power smart devices.

The problem of intelligent placement of relays to improve the performance of mobile DTNs has been studied [24–27]. [24] presents a scheme to deploy relays, called throwboxes, in mobile DTNs to maximize data rate between mobile nodes. [26] studies the hardware architecture for such relay nodes in an attempt to increase the lifetime. DistressNet deals with a slightly different problem where instead of maximizing the data rate between mobile nodes, we focus on optimizing the end-to-end aggregate throughput of all data flows in the network. Sink election is an important primitive in DistressNet. Recent solutions assume that sink is always fixed and concentrate on delay and energy consumption [28]. In our application, the position of sinks must be chosen based on traffic pattern.

DistressNet improves upon the above corpus in several ways – it provides services like file storage and social networking specifically designed to be available even in the presence of network delay and disruption. Several optimization algorithms, integrated with the system, improve the aggregate throughput by creating contact opportunities or by choosing a data aggregation point. The system has been designed and is being developed in consultation with real emergency response personnel. Several versions were built and deployed on commonly available hardware, completely battery powered.

3. DistressNet system design

In this section we describe the system design of DistressNet. A conceptual model of the disaster recovery process is first presented. Then, the responder requirements gathered from our interaction with first responders is presented, followed by a list of design principles behind DistressNet. Finally, the DistressNet network architecture is discussed.

3.1. Conceptual model

DistressNet addresses the needs of the disaster recovery process in the US&R area, as opposed to the medical triage area of a large disaster, as illustrated in Fig. 1. When a disaster hits an urban metropolitan area that spans tens of square miles (2011 Joplin tornado) or hundreds of square miles (2011 Japan earthquake), power and communication infrastructure are rendered unusable. A situation report about the 2011 Joplin Tornado [29], three days after the disaster, offers a glimpse into the situation: electric services are still being restored, a few cell sites have been restored, cell phones are being distributed and satellite telephone has been set up. In this kind of environment, presence of broadband internet access cannot be assumed, and blanketing a large urban area with battery powered communication hardware is near impossible. Providing data to US&R responders at high spatial and temporal resolution, with only tens of routers becomes a challenge. We assume that in such an environment, there are multiple collapsed buildings (buildings in Fig. 1) or rubble piles in the affected area ("Rubble Pile" in Fig. 1), and the Command/Control center ("C2" in Fig. 1) is situated tens of miles away from the affected area. Limited internet connectivity is available only at the C2. There is some mobility in the area ("Vehicle Node" in Fig. 1) as medical supplies and rescued victims are transported from the field to the C2.

3.2. US&R responder requirements

DistressNet was built over two years, based on inputs from first responders as well as iterative improvements from implementing various design choices. The FEMA equipment cache list [30] gives the reader an idea for the size, cost and bulk of equipment currently used by US&R teams. Based on this list and interaction with Texas Task Force 1 US&R we outline the responder requirements and their relation to computer network metrics.

3.2.1. Qualitative requirements

These requirements improve disaster response by reducing the time required for personnel to perform their tasks, by enhancing the quality of available data using new methodology or technologies and also by removing practical roadblocks.

REQ1: Smart victim detection under rubble: Highly sensitive seismic sensors that pick up vibration from a rubble pile were used during the 9/11 emergency to locate trapped victims [31]. First responders can listen for human voices or activity through attached headphones, and can locate them by asking victims to tap on nearby pipes.

However, the low frequency sound created by shaking buildings and nearby human activity interferes with this detection process [31]. Automatic noise filtering (1) eliminates the "All Quiet" condition required for seismic sensor use (which halts rescue efforts in the immediate surroundings); and (2) enables the re-deployment of on-site personnel to other areas of the disaster.

REQ2: Digitized building information: Whenever US&R teams search buildings, the search status of the building is indicated using markings (called X-codes, FEMA/INSARAG format) painted in day-glo orange on walls (tagging), for the benefit of other teams. This includes data like the last search date/time, presence of hazards etc. Digitizing such tags using low power motes will provide the C2 with high situational awareness due to the variety of information that can be sensed on motes. By digitizing building tags and enabling automated data collection, resources can be allocated by the command center more efficiently.

REQ3: Disconnected social networking: Social media like Twitter are increasingly being used by the public during the aftermath of disasters for communication and information dissemination [32]33. Information sharing by responders during disaster recovery could possibly enhance the recovery effort. However, there is no such service for disconnected networks like DTNs, since social media apps on smartphones assume the presence of internet connectivity. An equivalent service for DTNs will provide responders with an opportunity to share information without requiring constant connectivity, while automatically syncing with the Internet whenever Internet access is available.

3.2.2. Quantitative requirements

These requirements improve disaster response by improving existing metrics, such as aggregate network throughput and the time taken to detect separation in the team. In DistressNet, they are networking related solutions and improve metrics typically addressed by networking research.

REQ4: Fast team separation detection: During US&R operations in a collapsed building, team members may become separated from each other due to falling beams, or they may lose vital tools like cement saws accidentally. We present an algorithm that lets each team member know of any separation in the team independent of the team size, even when the separation or "cut" occurs many hops away. Team separation detection delay is measured in seconds. An app installed on a smartphone alerts a first responder immediately after a tool or team member is detected as missing, enabling recovery from the situation within seconds.

REQ5: Improved situational awareness: Situational awareness can be improved with a large amount of accurate data at a high temporal and spatial resolution being made available periodically with the least delay. However, this task is challenging because the network is fragmented due to the size of the area. Much of previous research has been devoted to improving network performance metrics like throughput, packet delivery ratio and delivery latency. We quantitatively measure situational awareness by aggregate throughput in Kbps and delivery latency in seconds. By placing additional hardware in the disaster affected area, the aggregate throughput of the network is increased and the end-to-end delay is decreased, providing the C2 with increased situational awareness.

3.3. Design principles

Based on first hand accounts of US&R deployments and responder requirements, we decided on a set of principles that shall govern our design of DistressNet. A list of applicable system design principles can be found in [34].

PRI1: Unmodified COTS devices: Governmental organizations are increasingly adopting COTS devices because of the available support and software, at fairly economical prices as compared to a custom platform. In many instances, US&R responders have used their own personal iPhones during disasters to email photographs of rubble piles. In any case, one cannot assume a "jailbroken" device where one can have complete control, as is fairly common with hardware platforms used in academic research. Instead, the system has to be designed such that stock capabilities of popular COTS devices are sufficient, so that devices can be borrowed and setup easily. A custom routing protocol or user replaceable batteries are not possible on the iPhone, as an example.

PRI2: Open standards and protocols: Standardized protocols, preferably of international scope, are emphasized. Certain WiFi channels are allowed in Japan but not in the USA; such issues should be planned for. At every layer of the system, open formats and widely supported protocols make integration of hardware with other international teams much easier.

PRI3: App oriented design: Because complexity is pushed towards the application layer, updating the system becomes easy and does not need recompiling/reflashing the entire device, especially during disasters. When deployed on a large scale over a variety of heterogeneous devices, PRI3 will significantly reduce roadblocks encountered in platform adoption. At the same time, simplicity in these complex apps is necessary: when a human–computer interface is present, having more than three buttons will cause the device to be left behind in a vehicle, instead of being used by first responders.

PRI4: "Premature optimization is the root of all evil": With DistressNet we first build a proof-of-concept implementation that captures most of the required functionality, and then iteratively optimize the system based on deployment experiences. For example, we trade performance for simplicity in the source routing optimization, by using a simplistic vehicle movement model. The gained simplicity makes it easier to deploy DistressNet as a whole with limited manpower, providing us with valuable experience which we can then use in the next iteration of the source routing protocol.

3.4. DistressNet hardware architecture

Here we present the hardware and network architecture of DistressNet, using the typical deployment scenario in Fig. 1 as reference. Based on the hardware

characteristics, components can be classified in three classes: A, B and C. Since each component performs a function related to its hardware capabilities, there are three software classes as well – these are shown below. A comprehensive listing of all components with their hardware classes and functionality is available in Table 1.

3.4.1. Class A: Sensing/monitoring

[DistressNet components] A *BTag Sensor* (BTag stands for "Building Tag") is a 802.15.4 based monitoring device which is attached to buildings externally. A *Seismic Sensor* (a Delsar Life Detector, as shown in Fig. 2) is a device which monitors rubble piles for vibrations.

[Hardware] These components are typically implemented on low power, battery powered platforms such as an EPIC mote. The data sensed may be either time critical or informational. Being heavily duty cycled, they are designed to last for several weeks with a single charge. 802.11 support is rarely found on these devices, with 802.15.4 or no networking being more common.

3.4.2. Class B: End user interactive

[DistressNet components] *Smartphones* are used in DistressNet to access data while in the field. A BTag App (Fig. 5 in the next subsection) is used to program BTag Sensors once they are deployed, with relevant information. Using the widely used iOS SDK, the iPod Touch was customized. However, we were limited to the application layer since the SDK does not allow non-trivial modifications to the operating system for security reasons. The fact that 802.11 IBSS mode was readily supported out of the box made us choose iOS over Android.

[Hardware] Smartphones and popular network centric consumer electronics like tablets which have networking capabilities, but have limited resources. These provide a rich interface to the data collected in the field, while also providing some functionality themselves. Not as resource constrained as sensors nodes, most devices have 802.11 capability and are designed to last a few days on a single charge. These devices also have various sensors.

3.4.3. Class C: Network backbone

[DistressNet components] *Sensor Proxies*, *Base Stations*, and *Data Waypoints* are class C components in DistressNet. They are portable, battery powered devices which provide basic wireless networking functionality and are deployed in the field. An example is a common 802.11 router found in most homes today. They can be assumed to have expansion ports to provide additional functionality like persistent storage or cellular connectivity. These can either be static or deployed inside a vehicle. In DistressNet, these devices are the only ones implementing DTN capabilities.

[Hardware] These components use the RB433UAH routerboard (Fig. 3). RB433UAH has 3 MiniPCI slots and 2 USB ports, allowing for two 802.11abgn wireless cards configured for 2.4 and 5 GHz. It also has 512 MB of NAND flash and 128 MB of RAM. OpenWRT is an open source operating system compatible with this router, which was chosen because of the openness and the wide range of software and support available. The USB port can be used to provide

Table 1
DistressNet components.

Component	Class	Function
BTag sensor	Class A	Sensing
Seismic sensor	Class A	Monitoring
Smartphone	Class B	End user interface
Data waypoint	Class C	Networking
Base station	Class C	Networking
Vehicle node	Class C + A	Networking & proxy
Sensor proxy	Class C + A	Networking & proxy

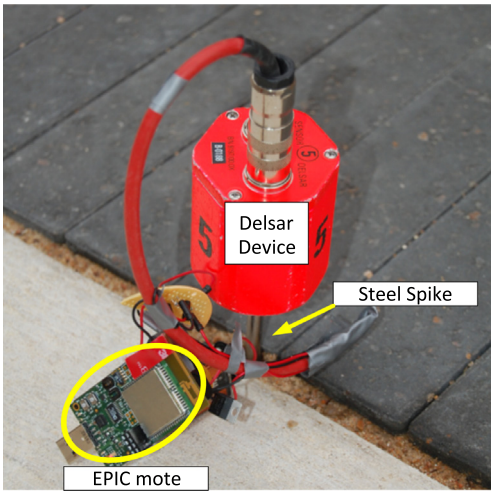


Fig. 2. Delsar life detector with steel spike driven into the ground, and interfaced with an EPIC mote.

Fig. 3. Mikrotik RB433UAH wireless router mounted on a tripod and powered by a battery.

functionality like a new physical layer such as 802.15.4 (used in Sensor Proxies), enhanced storage like a USB flash drive (for Data Waypoints) or both.

3.5. DistressNet software architecture

We briefly describe the software used in the three device classes of DistressNet.

3.5.1. Class A: Sensing/monitoring

The applications deployed on these devices include sensing and services (BTag, Sink Election and Sensing in Fig. 4). When networked, these components use a traditional stack like UDP over IPv4 or v6. In DistressNet, we use RPL, an IPv6 Routing Protocol for Low power and Lossy Networks, as the default routing protocol. To utilize the DTN capabilities of DistressNet, these devices need a Sensor Proxy which is a Class C device.

3.5.2. Class B: End user interactive

Such devices are typically incapable of routing or advanced networking capabilities and have limited, but not scarce, resources. An example is a smartphone or a tablet which we can install apps on. While users do not have access to the OS or lower layers, they can still use a nearby router as a proxy for their networking needs. Since an iPod Touch does not allow applications (developed using the offered SDK) to modify networking routes in the OS for security reasons, it is restricted to one hop communication. If the smartphone needs to utilize DTN capabilities, it sends its data to a nearby Sensor Proxy.

Apps installed on these devices are more of data consumers than data generators. The file sharing and social networking apps (as shown in Fig. 5) are end user interfaces for accessing the Fog. The network stack on these devices consists of TCP/UDP over IPv4/v6 and 802.11. Additional non-network software includes device drivers for cameras and various sensors, and apps which use these sensors.

3.5.3. Class C: Network backbone

The dominant stack used in DistressNet is 802.11abgn in IBSS mode below IPv6/v4 and UDP (Fig. 6). For implementing DTN functionality, we used the IBR-DTN implementation which is readily available as a package for OpenWRT. The software ecosystem offered by OpenWRT allowed us to completely customize the router according to our needs, with a variety of available and possible functionality. DTN is implemented as an overlay network of nodes where multiple local clients can connect to a local DTN server (Bundle server in Fig. 6) in the application layer. A special DTN app on the router which can talk to the Class A device as well as the DTN server and has 802.11 connectivity provides DTN proxying functionality for class A devices ("802.11–802.15.4 Switch" in Fig. 6). Class C devices perform two distinct classes of routing: DTN specific routing and WMN specific routing. While OLSR was used for mesh routing because of the stable implementation, there are several routing protocols specifically designed for both opportunistic and scheduled delay tolerant networks. Epidemic routing and PRoPHET routing are two popular protocols which we consider. Our waypoint placement algorithm also affords a simple source routing protocol for DTNs, as discussed in future sections. Since most COTS WiFi compliant devices belonging to class B support only the 2.4 GHz band, we decided to use the 5 GHz interface exclusively for DTN routing, mesh routing and DTN services which are unique to class C. DHCP is provided on the 2.4 GHz interface for clients to connect. All routers have statically assigned IPs – router n has an IP of

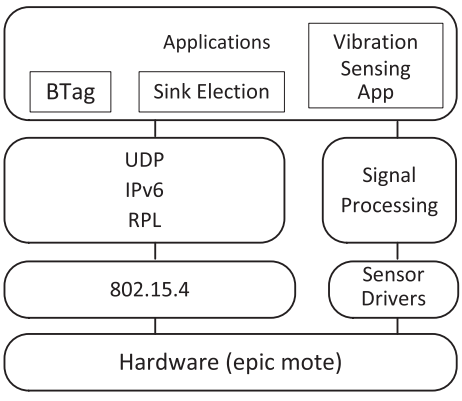


Fig. 4. DistressNet software architecture: Class A: Sensing.

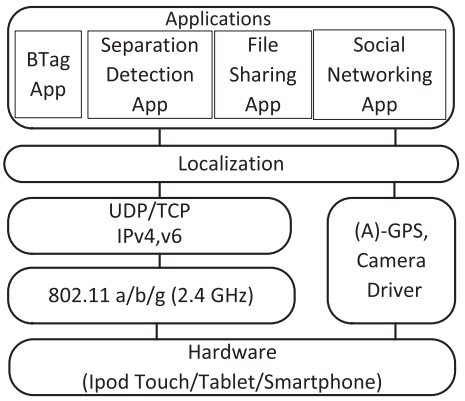


Fig. 5. DistressNet software architecture: Class B: End user interactive.

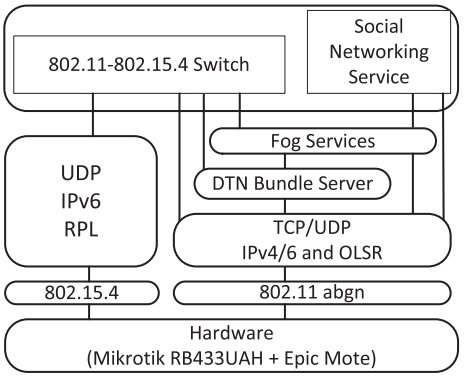


Fig. 6. DistressNet software architecture: Class C: Network backbone.

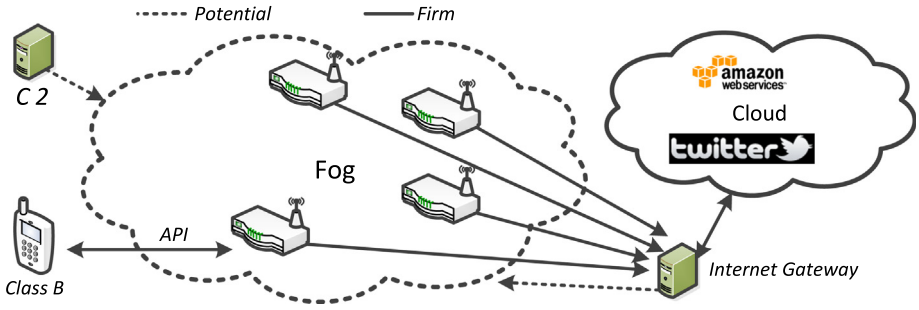


Fig. 7. Firm and potential flows in DistressNet.

192.168.50.*n* for its 5 GHz interface and 192.168.*n*.1 for the 2.4 GHz interface. Each router can handle 255 end user devices – they are assigned IPs in the 192.168.*n*.0/24 range.

A "Bundle" (RFC 5050) is the primary data unit in DistressNet DTN. Each DistressNet DTN node is identified by a URI like *dtn://dn.zigbeegateway*1. A client application with an ID of *mote*1 can connect via an API to the local "Bundle Server". Then, any traffic intended for this app will simply need to be addressed to *dtn://dn.zigbeegateway*1/*mote*1. Examples of functions provided by the bundle server API includes registering the application name (e.g., "mote1"), setting a destination, requesting encryption or authentication or custody transfer and setting the lifetime. All communication in the DTN layer of DistressNet can be encrypted and authenticated, as defined in RFC 6527. Each DTN node first generates, pre-deployment, a 2048-bit RSA public/private key pair. The public keys of all nodes are aggregated and shared among all DistressNet DTN nodes. This data is used for bundle encryption in a public-key cryptography fashion. It is to be noted that bundle encryption is always between source–destination and authentication is always on a single hop basis. Bundle authentication in DistressNet uses the HMAC-SHA1 message authentication cipher, which encrypts a message based on a key. In this case, the key is a pre-shared plaintext key of arbitrary length that is different from the public/private keys.

4. DistressNet applications

In this section we present a limited set of DistressNet applications: the *vibration sensing app* (as shown in Fig. 4), the *BTag*, *file sharing*, *separation detection* and *social networking apps* (as shown in Fig. 5), and the *social networking service* and *Fog Services* (as shown in Fig. 6). Before we proceed, we first define how data is created and used in DistressNet using the concept of flows. A flow defines the source and destination for a particular type of data, either deterministically or probabilistically. Since data can be sent using various paths in a DTN, the optimization of data movement in the network is critical. The algorithms and mathematical framework for Fog optimization is discussed in Section 5.

4.1. Data production and consumption

The main idea behind servicing DistressNet applications, as shown in Fig. 7, is to mirror a traditional external service (like Flickr, Amazon S3 or Twitter) in a local DTN, containing a mixture of mobile and static vehicles and devices. In DistressNet, mobile teams visit multiple *points of interest* in the affected area. Devices may be deployed at various *locations/points of interest* for unsupervised monitoring of the environment. These points of interest eventually act as *sources* and *destinations* of data in DistressNet. As shown in Fig. 7, special points of interest in DistressNet are the *C2*, and an *Internet Gateway* which may not be co-located with the C2. The Internet Gateway provides access to "Cloud" services, which are common in traditional internetworks and becoming of interest to emergency response applications. The link between DistressNet and Cloud services, however, is a high cost link. Internet access for large amounts of data may be high cost and resource demanding. *Therefore, and this is a key idea in DistressNet, remote Cloud's services are provided locally in the Fog (i.e., the instantiation of a Cloud in the intermittently connected DistressNet).* Because accessing services directly from the Cloud incurs a high cost, if at all possible, it is much more efficient to instead access services from the Fog.

As shown in Fig. 7, in DistressNet there are several data "flows". Some flows are "firm" – these have a pre-decided destination and source. Other flows are "potential". These potential flows represent data to be stored somewhere in the Fog, without a specific destination. For the data stored in the Fog, DistressNet uses an "availability" metric which ranges from 0% to 100% and denotes the importance of the data. An availability of 100% means that data will be available on all Fog devices in FogNet, whereas an availability of 25% means that data will be available on at most a quarter of Fog devices. More critical data will have higher availability. Examples of data and its availability metric include data generated by BTag Sensors, which is not critical and has an availability of 33% (i.e., the BTag Sensor Data will be probabilistically stored on about 30% of existing destinations in the Fog), and data generated by Seismic Sensors which is critical and requires 100% availability (i.e., the Seismic Sensor Data will be stored on all existing destinations in the Fog). The specific data flows in DistressNet, and their availability metric are discussed in the following subsections.

4.2. File sharing app (Class B, REQ3)

The file sharing application allows authenticated users to share data with other users, or groups of users. As an

example, one can imagine a team of responders sharing the layout of an explored building along with current hazards, with other nearby teams. An important feature that emphasizes the need for such a service is that the destination for data sent is unknown – it is simply stored in the "Fog" and accessed by anyone who connects to FogNet, via a class C device.

The client application runs on a Class B user interface device which connects to a nearby Class C device. The *Fog Services* server application, running on a Class C device, as shown in Fig. 6, handles the authentication. Users of this application can ADD/DELETE/MODIFY files which they own. They can also share files with other teams. Examples of files that users can upload include video taken using a smartphone's camera. Users can also specify whether they want to backup these files to an external storage provider like Amazon S3 or Flickr. If a client wishes to use their own external account, an encrypted query, using the provider's API (S3's API or Flickr's API), is sent to the Internet Gateway. Clients can, in addition, specify availability metrics depending on the importance and criticality of the data. *The flows for file sharing are: for each class C device implementing Fog Services, one firm flow to the Internet Gateway for backup, and one potential flow from the Internet gateway to the Fog for synchronization.*

4.3. Social networking app (Class B, REQ3)

This application uses the aforementioned file sharing service to provide additional services like inter-team messaging, and allows victims to publicly tweet their status. A *social networking app* running on a class B device connects to a nearby class C device which offers the *social networking service* (i.e., possibly running on a Vehicle Node, for example). Users can communicate with other users, or mass message other teams or team members. They can also choose to post messages to an external Twitter account, for example. It is important to note that such messages will be available in the Fog, as well as on their Twitter profiles – thus, an external service like Twitter is mirrored in the Fog. All data from Twitter is pulled regularly by the Internet Gateway and sent to the Fog (Fig. 8) so that the data stays synchronized. *The flows for social networking are: For each class C device implementing social networking service, one firm flow to the Internet Gateway for backup, one firm flow to the C2 for situational awareness, and one potential flow from the Internet Gateway to the Fog for synchronization.*

4.4. Vibration sensing app (Class A, REQ1)

The FEMA US&R equipment cache list [6] mentions Delsar Life Detection sensors: a steel spike that is driven into rubble which responders can then monitor for voices or knocks from victims. Upon manually probing the rubble at different places, the victim can be localized and rescue operations can commence. Since these sensors do not have any native networking capabilities, EPIC motes are used to provide an interface, creating a *Seismic Sensor* component. However, there are sources of noise like footsteps and vibration from nearby vehicles which are also picked up. The goal of the vibration sensing app is to automatically detect and classify the source of vibration. To profile these sources the steel spike of the sensor was driven into a small wedge in a pavement outside our building on campus. Three sources of noise/data were profiled: a stone dropped from a height, footsteps of pedestrians and a knock made by a hammer on a pavement. The fixed-point in-place 1024-bin FFT is shown in Fig. 9.

Algorithm 1. k-NN Classifier

1: **for** each $s_i \in s_1 \ldots s_{gn}$ **do**
2: Compute $d_i \leftarrow \sqrt{(F_1 - s_i^{f_1})^2 + (F_2 - s_i^{f_2})^2}$
3: **end for**
4: $r_1 \ldots r_k \leftarrow$ The k smallest d_i
5: *groups* $\leftarrow$ Union of groups that each of $r_1 \ldots r_k$ belong to
6: $G \leftarrow$ most common group in *groups*

It is important to note that the amplitude of the signal alone cannot be used to classify a source. Hence, two features were extracted from the FFT: (i) average value of the frequencies weighted by their respective amplitudes (f_1) and (ii) the mean amplitude of the frequencies (f_2). We then used these features in a simple KNN (k-nearest neighbor) classifier, motivated by the fact that classification has to be done on a resource constrained *Sensor Proxy*. Suppose that we have g different types of data $G_1 \ldots G_g$ and n samples for each group, for a total of gn samples $s_1 \ldots s_{gn}$. Let each sample be a vector consisting of two features $[f_1 f_2]$. The KNN classifier first needs to be trained using these samples. Training consists of storing each sample and its corresponding group in memory. Now, given a new sample $S = [F_1 F_2]$ that needs to be classified, Algorithm 1 can be used to calculate the group G that S belongs to, thus identifying the source of the vibration. A Seismic Sensor which contains the vibration sensing app communicates with a nearby Sensor Proxy, which also acts as an entrypoint into the Fog. *The flows for Vibration Sensing are: for each Sensor Proxy, one firm flow to the Internet Gateway for backup and one firm flow to the C2 for situational awareness.*

4.5. Building monitoring (Class A, REQ2)

BTag Sensors are low power devices which manage metadata related to a building from a search and rescue viewpoint. The primary motivation for this component was the current state of art, where US&R personnel use paint on walls (Fig. 10(a)) to store information about the current search status of a structure. This includes information like the number of survivors inside, the location of chemical hazards if any and the most recent date/time that the structure was searched [35]. This information is most likely to remain constant and not change very often. Any vehicles in the vicinity which drive by can electronically gather data from the BTag Sensors, especially if they are outfitted with special chemical or air quality sensors.

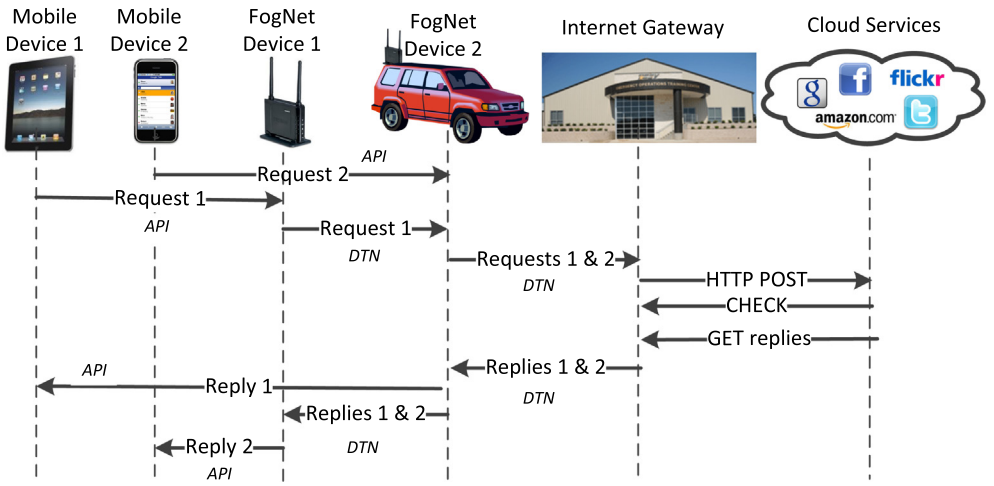


Fig. 8. Sequence diagram for accessing cloud and social networking services (e.g., Twitter).

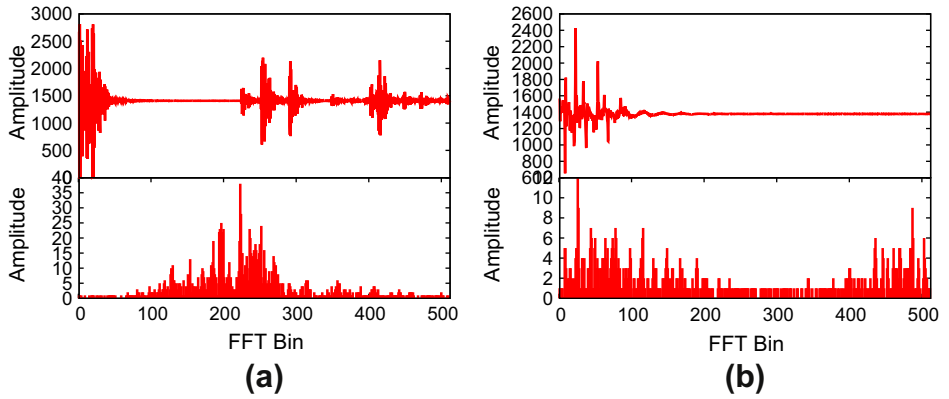


Fig. 9. Spectrum and signal of (a) stone drop (b) footstep.

For the purposes of saving energy and choosing an aggregator who is most likely to encounter a vehicle node, these sensors perform *Sink Election*. Such aggregation by a sink ensures that nodes which may not have a LoS to a nearby road can still communicate their data to the C2 efficiently. These tags are first programmed by search and rescue personnel once the search is complete, by using the "BTag App" (Fig. 5). A screenshot of the app running on an iPod Touch is shown in Fig. 10(b). *The flows for Building Monitoring are: for each Sensor Proxy, one firm flow to the C2 for situational awareness.*

4.6. US&R team separation detection app (Class B, REQ4)

US&R operations in an unexplored large areas with low visibility and potential hazards (e.g., collapsed tunnel, chemical spills) are dangerous. Any incidents involving team member separation or loss of vital tools (a "cut" in the network) can slow down the victim rescue process because of unnecessary delays. To meet the need for a separation detection method, we develop an iOS application on an iPod touch that enables each team member to monitor connectivity to a team leader *multiple hops away*, and warns a team member of physical separation from the team leader.

This app ("Separation Detection App" in Fig. 5) is inspired by the distributed cut detection algorithm presented in [36]. A node in an electrical network containing a current source will see a change in its potential when there is a partition in the network, enabling it to detect changes in network topology. Similarly, every node n in a computer network maintains a positive scalar value called the "state" $st(n)$, updating it using the formula $st(n) = (\sum_{N(n)} st(i))/(|N(n)| + 1)$, where $N(n)$ is the set of one hop neighbors of node n. A special source node S injects a value I by updating its own state using the formula $st(S) = \frac{I+\sum_{N(S)} st(i)}{|N(S)|+1}$. The state of each node converges, given a network topology. A node in the same partition as S after a cut will see its state converge to a higher value, otherwise it drops to zero. The convergence time is fast and the

Fig. 10. (a) Markings indicate that the buildings have been searched; (b) screenshot of a BTag app on an iPod touch.

maximum delay in experiencing such a change is bounded, as shown in [36]. Thus, this algorithm helps first responders detect accidental separation in the team by using only one hop communication.

5. DistressNet services

In this section we describe services and algorithms which optimize and implement the above functionality. FogNet refers to the use of potential and firm flows to build a DTN that enables file sharing (Fog Services in Fig. 6) and social networking (social networking service in Fig. 6). In order to optimize these flows, Data Waypoints need to be placed at certain places in the area of deployment. The aggregate throughput obtained by placing these waypoints depends on the per-contact data transfer capability of nodes, which is a function of the node's speed and payload size. In addition, BTag and Seismic Sensors which do not have a Sensor Proxy nearby can send data directly to a Vehicle Node. In either case, those nodes need to perform Sink Election in order to elect a data aggregator.

5.1. FogNet services

FogNet enables file storage and social networking in a disconnected DTN, and is built as a DTN overlay. Such services enable teams of responders to store important data on nodes, access them on demand and share them with other teams. A traditional cloud storage service has the following functionality and properties: (i) The ability for a client to upload a file to the cloud, without specifying a destination node; (ii) The ability for a client to retrieve a file, without specifying the location of the file; and (iii) Data robustness due to intelligent replication performed by the cloud service back end. *We term the same Cloud functionality when provided over a DTN as a "***Fog***" service.*

FogNet clients are class B devices, while the FogNet services are implemented on class C devices. Sources and destinations of flows in DistressNet belong to classes A or B, but never class C. The file sharing and social networking app on class B devices accesses Fog Services and social networking service on class C devices, respectively. The social networking service in turn uses the Fog Services for storage of data. The "Fog Services" app on class C devices implements the ADD/DELETE/MODIFY actions that can be performed by clients. Whenever a client performs any of these actions, the server performs local modifications. These local modifications are then communicated via DTN to other FogNet devices. Thus, actions performed in the Fog translate to traffic created over DTN.

5.1.1. FogNet API and security issues

The FogNet API insulates the user, first responders in this case, from the inner workings of the underlying DTN. It aims to provide a service similar to those offered by cloud storage services like Dropbox and Amazon S3. Three primitives called ADD, DELETE and MODIFY provide an interface into FogNet. The ADD primitive adds a file into the Fog. This file could consist of a tweet as produced by the social networking app, or a photo from the user's phone as produced by the file sharing app. When an ADD request is sent from a Class B smartphone to a nearby Class C router (over raw UDP or HTTP PUT over TCP for example), the Class C router first marshals the data into a bundle. This bundle is replicated on multiple Fog devices according to the availability of the associated flow. The actual Fog devices are chosen based on the output of the optimization problem presented in the following section. The bundle is

source routed to those devices, again based on the solution to the optimization problem.

The MODIFY/DELETE primitives are similar in behavior. A MODIFY operation causes the Class C router to marshal the *difference* between the current version of the stored file and the new incoming file into a bundle. This bundle is replicated to the same Fog devices which contain the original. The Fog devices will locally modify their copy upon receipt of the bundle and push it to the user's Class B device when he/she connects. The DELETE operation simply sends a bundle to the Fog device which says that the local copy of the file on the device should be deleted. When a user connects to a Class C device which has received a bundle, the app on the smartphone will delete its local copy. The synchronization between Class B and Class C devices is very simple. When a user's Class B device discovers a Class C router nearby, it can supply a list of files stored locally and ask for changes to those files. The Class C router will reply with a list of changes, which the user's device can apply to its local copy. This way, any app on the smartphone can use the primitives mentioned above to use the DTN as a cloud storage service.

[External accounts] The above primitives are for those users who want to use FogNet's social networking services. If a user prefers to use their own personal Twitter account or in general any service with a RESTful API, it is possible. FogNet is capable of relaying data streams with end to end encryption, using the simple trick of marshaling HTTP requests into bundles and transferring them over the DTN. Users need not disclose their existing external credentials in order to use FogNet.

We consider Amazon S3 as an example of an external RESTful cloud storage service provider. Files are uploaded to S3's servers using a published API which offers both a REST and a SOAP interface using XML. When a user signs up, a secret key is assigned. This secret key is then used alongside HMAC-SHA1 in order to authenticate all HTTP (optionally, HTTPS) requests. Whenever a user wishes to place a file in S3, a challenge string is first constructed based on a predefined ruleset, which then serves as the "message" in HMAC-SHA1. The output, which is a base64 encoded string, forms part of the HTTP request header. The entire HTTP header and body (if applicable) is then sent to Amazon by the client. This HTTP header and body is intercepted by the Fog Service and sent to the Internet Gateway via DTN, after being marshaled into a bundle.

5.2. FogNet service internals and optimizations (REQ5)

Mobile vehicles in a disaster stricken area can be leveraged as data carriers. They can gather data from various data sources such as sensor sub-networks and deliver it to data consumers like the C2 and/or a group of first responders in a building along the path of the vehicle. Some vehicles like ambulances go to the C2 more frequently, while there may be some vehicles that do not visit C2 for hours. Time is a critical factor in disaster response. Thus, US&R operations happen simultaneously all around the disaster area, as long as the access to the area is cleared. These operations typically last for days; such patterns indicate that we may be able to take advantage of the (planned) paths of vehicles in order to move data more efficiently in DistressNet. Teams are assigned to search areas by a dispatch command. Our assumptions about this model include the fact that vehicles always move in loops at a fairly constant speed, that all devices in the model have enough on board storage (expandable via USB, for example), and that vehicles do not arbitrarily change paths or schedules. The mobility model in DistressNet can be thought of as a reduced version of the Post Disaster Mobility (PDM) Model [37]. The authors of [37] model the mobility during the disaster recovery process in terms of "Centers" which are points of interest in the area, "Mobility Patterns" which include event-driven/Center-to-Center and "Mobile Agents" which include rescue workers, police patrol cars and ambulances. We have chosen to model only the "Cyclic route" mobility pattern for simplicity and tractability – incorporating all of PDM model's mobility patterns is left as future work. Note that the movement of responders around a point of interest is not modeled – only the movement of vehicles in the disaster area is considered. This is because responders are assumed to move only around buildings and thus, do not contribute to the data muling process in the DTN. However, vehicles which roam the area are equipped with routers and do participate in the data muling process.

[Importance of payload size] The Data Transferred per Contact (DTC) is the amount of data transferred between a stationary node and a mobile one, either over 802.11 or 802.15.4. The DTC depends on many factors including the size of each packet as well as the speed of the vehicle. It is therefore important to consider these factors as variables when modeling DistressNet.

[Preliminaries] Consider n vehicles $V = \{V_1 \dots V_n\}$ in DistressNet, with the path of each vehicle being a loop and hence representable by a closed polygon. For a vehicle $v \in V$, let this polygon be called $Path(v)$, the speed of the vehicle be $Speed(v)$, the time taken by a vehicle to go from point A to point B both on $Path(v)$ and along it, be:

$$T(A, B, Path(v)) = Dist(A, B, Path(v))/Speed(v)$$

Let the set of data sources be S and the set of destinations, D.

[Flows] Any deployment additionally has a set of firm flows F, with each flow $F_i \in F$ having a data source $F_i^{src} \in S$, a destination $F_i^{dst} \in D$ and the size of the data F_i^{data} that can be sent from the source to the destination. Note that a node may act as a source as well as a destination in different flows. A similar set of potential flows P is also defined, but each potential flow $P_i \in P$ has only a source $P_i^{src} \in S$, an availability $0 < P_i^{avail} < 1$, the maximum data size P_i^{data} but no destination. We construct the set of modified potential flows Q, from P such that each flow $P_i \in P$ is assigned each destination $d \in D$. Let $Z \subseteq (F \cup Q)$ be the final set of selected flows such that:

- Every firm flow is included, i.e., $F \subseteq Z$
- The availability of each potential flow is satisfied. Mathematically, for every $P_i \in P$, there are at least $|D| \times P_i^{avail}$ flows in Z, for which the source is P_i^{src}, chosen from Q.

[Waypoints] A "waypoint" is a router placed at the intersection of the paths of two vehicles $v,w \in V$ such that data can be dropped by v and picked up by w or vice versa. Let X be the set of all possible waypoint locations (which is where ever the paths of any two vehicles intersect). A "solution set" for each flow $Z_i \in Z$ means a sequence of alternating vehicles and data waypoints that are capable of carrying data from the source to the destination, i.e., a set:

$$\{Z_i^{src}, v_i^1, x_i^1, v_i^2, x_i^2 \ldots Z_i^{dst}\}, v_i \in V \quad \text{and } x_i \in X$$

The physical travel delay of a packet in a DTN is a major component of the delivery delay, assuming other factors like the queuing delay at a router, and the time between packet generation and pickup up a vehicle are negligible. The travel delay for a solution set $\{Z_i^{src}, v_i^1, x_i^1, v_i^2, x_i^2 \ldots Z_i^{dst}\}$ is the sum of travel delays between Z_i^{src} and x_i^1, between x_i^1 and x_i^2 etc. Therefore, the time taken for data to flow using the solution set for a flow Z_i will then be:

$$T(Z_i) = T(Z_i^{src}, x_i^1, Path(v_i^1)) + \cdots + T(x_i^n, Z_i^{dst}, Path(v_i^n))$$

[DTC] The maximum size of data that can be transferred on a flow depends upon the DTCs of each contact in its solution set: Z_i^{data} can now be defined as

$$Z_i^{data} = min(contact(v_i^1), contact(v_i^2) \ldots contact(v_i^n))$$

where $contact(v)$ is the maximal DTC that is possible between a node and the vehicle v traveling at a speed $Speed(v)$.

[Problem formulation] The Waypoint Placement problem is now defined as follows – given an upper bound X_{max} on the number of waypoints (e.g., limited available hardware), we choose $X^* \subseteq X$ such that:

- For each flow $Z_i \in Z$, the solution set contains vehicles which are found in V and waypoint locations which can be found in X^*.
- The aggregate throughput $\sum_{z \in Z} \frac{Z_i^{data}}{T(z)}$ is maximized.
- Optionally, the cardinality of X^* is less than X_{max}.

To efficiently solve the above problem, we first create a representative graph G. To construct this graph, first create a vertex for each unique source and unique destination. Next, create a vertex for every possible waypoint location $x \in X$. Draw an edge between any two vertices m_i and m_j whenever a single vehicle v passes through both the vertices. The key intuition here is that the weight of this directed edge is nothing but the time taken by the vehicle v to physically transport the data from m_i to m_j. In order to account for uncertainty in the mobility model, a quantity $0 < arr(v) < \frac{Path(v)}{Speed(v)}$ is added to the travel delay. Thus, the weight of the edge in question will then be $arr(v) + Time(m_i, m_j, Path(v))$. $T(Z_i)$ can then be modified to take this into account by adding $arr(v)$ to each term appropriately.

Let a binary selection vector $\mathbf{c} = [c_0, c_1, \ldots, c_{-X-}]$ denote whether a possible location $x_i \in X$ is chosen to be a data waypoint (1) or not (0). Let the subgraph G^* denote the graph formed by G by removing vertices indexed by a 0 in $\mathbf{c}$. The problem is now to find a binary vector $\mathbf{c}$ such that, operating on G^* we:

$$\text{maximize} \sum_{i=0}^{|Z|} \frac{Z_i^{data}}{T(Z_i)} \tag{1}$$

$$\text{subject to } T(Z_i) \neq \infty \tag{2}$$

$$\sum_{i=0}^{|X|} c_i \leqslant X_{max} \tag{3}$$

Constraint (2) ensures that there is always a path in G^* between the source and destination for each flow. This is because the delay for a non-existent edge will be set to inf, or equivalently, G^* can be made fully connected with the newly created edges having a very large weight. Constraint 3 ensures that the number of waypoints deployed is less than or equal to the maximum possible. This problem can be recognized as a binary integer programming problem and is thus NP-hard. Popular heuristics include the branch-and-bound algorithm, which is available in MATLAB as `bintprog` or various algorithms available in ILOG/CPLEX. Once the selection vector is available, **source routing** can be performed by building the optimal path for each flow in G^* and noting the waypoints used in that flow.

[A polynomial heuristic] can be admitted if X_{max} is unbounded. The heuristic in Algorithm 2 chooses $\mathbf{c}$ such that the aggregate throughput is maximum. First, all simple paths between the source and destination for each flow is computed using a depth first search algorithm (Step 3). For each path, the throughput is computed by dividing the time taken for a path (sum of edges) into the maximum data that can be transferred on that flow, by considering per-contact data transfer. The path with maximum throughput is then chosen (Step 4). All vertices in this path excluding the source and destination are added to a previously empty set X^* (Steps 5, 6). This process is repeated for each flow. The set X^* will then contain the set of vertices which will maximize the aggregate throughput by maximizing the throughput for each flow, given that X_{max} is unbounded.

Algorithm 2. Polynomial Placer

1: $X^* \leftarrow \phi$
2: **for** each $Z_i \in Z$ **do**
3: $paths \leftarrow$ depth-first-search in G between Z_i^{src} and Z_i^{dst}
4: $path \leftarrow$ the path in $paths$ with maximum $\frac{Z_i^{data}}{T(Z_i)}$
5: Add all vertices in $path$ to X^*
6: Remove Z_i^{src} and Z_i^{dst} from X^*
7: **end for**
8: $\mathbf{c} \leftarrow \mathbf{0}$
9: **for** each $x_i \in X$ **do**
10: **if** $x_i \in X^*$ **then**
11: $c_i \leftarrow 1$
12: **end if**
13: **end for**

[Construction of Z] The process of constructing Z according to the constraints specified above is itself an optimization problem. To achieve a true globally optimal solution, every possible Z should be constructed and applied to the throughput optimization problem above. A binary integer programming model can do this exhaustive search much more efficiently. If the time and resource complexity cannot be afforded, any combination of Z can be tried till a suitable placement of waypoints is obtained.

[User input] In order to use DistressNet and use FogNet service optimization, a GIS map of the affected area is needed, along with the locations of important points of interest like the C2 and collapsed rubble. Approximate paths of the scheduled patrols of each vehicle in the area, as well as their speed is required. The flows in the network and their availabilities are required. An availability of 1 for a BTag data flow is the equivalent of a functional requirement like "Digitized X-Codes are very important and should be available at all devices in the Fog at all times". Based on this set of information, chosen waypoint locations will be provided to the responders by the optimization program and Class C routers will need to be deployed at these locations. It is to be noted that the aforementioned Fog service will not cease to function in case the user input data like the speed of vehicles suddenly changes. The performance of the Fog will be less than optimal, but since the system cannot control the movement of vehicles, there is little that can be done.

5.2.1. Sink election for sensors

Typical sources in the data management problem include BTag Sensors deployed around a building, for example. Not all of the senors in this sensor subnetwork may have access to passing vehicles due to the topology. It is therefore necessary to choose a sensor from this subnetwork to act as a sink which can act as a source of data by offloading aggregated data to vehicles. The most efficient sink can be selected globally by using a virtual vertex technique: in graph G, model a source vertex S as having access to a set of vehicles which is the union of the set of vehicles accessible by each sensor in the sub sensor network. This way, once the optimal paths are selected by searching for G^* (and hence the optimal vehicle for S), simply select a sensor which has access to this particular vehicle.

[A "Zero Cost" solution] that solves the problem suboptimally is necessary since a globally optimal solution cannot be possible pre-deployment, and also since gathering vehicular incidence data causes delays. The solution is simple: a sensor which has access to the most number of unique vehicles is given higher priority during sink election. This way, we simply increase the chance of the above algorithms finding a high throughput G^*, since the number of edges in G will be greater. The zero cost refers to the fact that we make use of RPL's routing tree structure to perform sink election and resolve conflicts among candidates. This ensures that there is no additional messaging overhead (e.g., for neighbor discovery), while at the same time providing unicast communication in the subnetwork.

Algorithm 3. Sink Election

```
1: procedure SINKELEC(For node i)
2:   sc(sink) ← 0
3:   if ScoreTimerFired then
4:     Calculate sc(m_i)
5:     if sc(m_i) > sc(sink) then
6:       Broadcast sc(m_i) to DAGRoot
7:     end if
8:   end if
9: end procedure
10: procedure SINKELECROOT(For the DAGRoot)
11:   if ElecTimerFired then
12:     sink ← id(Max(received scores))
13:     Broadcast sink to all children
14:   end if
15: end procedure
```

Mathematically, a single "source" in the previous section can be modeled as a set $M = \{m_1 \dots m_n\}$ of deployed sensor nodes. We assume that two sources are not connected; otherwise they can be modeled as a single source. Let $batt(m_i)$ be the residual battery charge of node m_i, $beac(m_i)$ the number of beacons received from vehicles passing by, and $uniq(beac(m_i))$ be the number of unique vehicles encountered by m_i. Now, Let $sc(m_i) > 0$ be a scoring function that assigns the eligibility score to any node m_i based on the above three parameters. The sink selection problem can be formulated as the choosing of a particular m_i such that we:

$$\text{maximize } sc(m_i) \wedge uniq(beac(m_i)) \tag{4}$$

$$\text{subject to } beac(m_i) > 0 \tag{5}$$

$$batt(m_i) > 0 \tag{6}$$

Algorithm 3 shows the distributed sink election using RPL's tree structure. The scoring function sc used was $sc(m_i) = 3 \times uniq(beac(m_i)) + 2 \times beac(m_i) + batt(m_i)$. This simple, computationally inexpensive function gives more weight to the number of unique vehicles seen by the node, while at the same time preferring nodes which frequently see vehicles. Nodes, after computing their score regularly (Step 4) and comparing it with the current sink's score (Step 5), send the score to the root of the DAG structure built by RPL. Because frequent sink changes are considered resource intensive, the root conducts election only at specified intervals. During this election process, potential candidates are compared and the node with the highest score is chosen (Step 12). In case of a tie, the node with the lowest id is chosen. The identity of the new sink is then broadcast to each child node, till the leaf nodes are notified as well (Step 13).

6. Evaluation

DistressNet is a complex wireless, sensor, adhoc and delay tolerant network system. In this section we present the performance evaluation of the individual components of

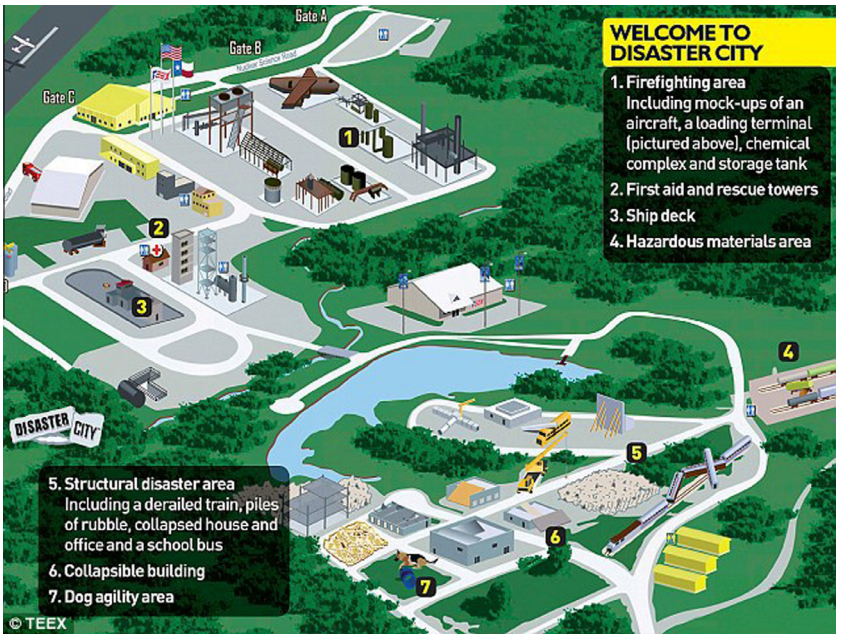


Fig. 11. Schematic showing the layout of Disaster City.

the system, as well as the entire system. The accuracy of the Vibration Sensing app has been evaluated in Disaster City (Fig. 11), which is a comprehensive 52-acre training facility for emergency responders with extremely realistic wrecks, including several rubble piles of wood and concrete.

First, we describe how DistressNet can be deployed in Disaster City, following a hypothetical disaster. The deployment effort needed from first responders, as well as the data needed from the deployment area is discussed. Next, the software stack used in the DistressNet device classes of A, B and C is presented. Finally, we progressively evaluate the system and the service internals by first finding the optimal payload size for 802.11 and 802.15.4 transfers for a given speed. Then, after verifying the correctness of the Sink Election algorithm using EPIC motes, we perform a simple experiment (Expt 1) which does not attempt to optimize the DTC. A simulation helps us choose the best routing protocol for this experiment. In a second deployment (Expt 2) we then show the improvement in aggregate goodput even in the presence of DTN security overhead, by the use of optimal payload size, source routing and optimal Data Waypoint placement. These experiments involved three vehicles and various data sources, and was conducted on our campus. The accuracy of the vibration sensing app w.r.t classifying the source of noise is shown. We finally share the lessons we learned during designing, building and evaluating DistressNet.

6.1. Deployment scenario

DistressNet was deployed in Disaster City during Summer 2012 as part of an exercise involving first responders. The objective of this deployment was to test the practicality of deploying DistressNet during a disaster, and not to test the performance of DistressNet over many days. At 10 A.M. on May 17, the Incident Commander convened a meeting at the Command Post (EOC in Fig. 12). Participants consisted of various groups specializing in technology such as UAVs, ground robots, marine robots and delay tolerant networking. The situation summary was as follows:

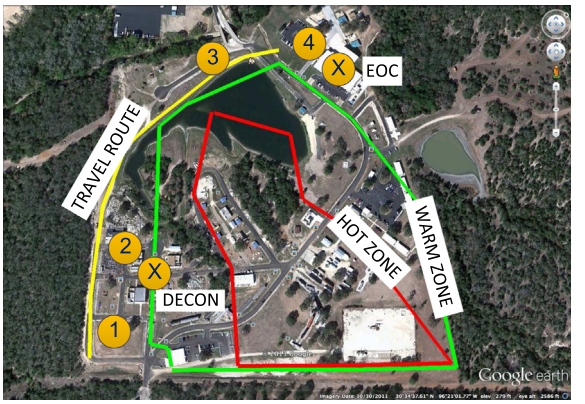


Fig. 12. Map of DistressNet deployment at Disaster City during Summer 2012.

> A major train derailment involving an estimated 12 cars of a freight train has occurred in the SE sector of Disaster City, TX. The derailment is adjacent to a producing oil well, underground oil distribution line, above and below ground power lines, commercial and private residences. Unknown at this time: damage to utilities and infrastructure but power is out in the area. Unknown if any car (s) are leaking but residents in the area report

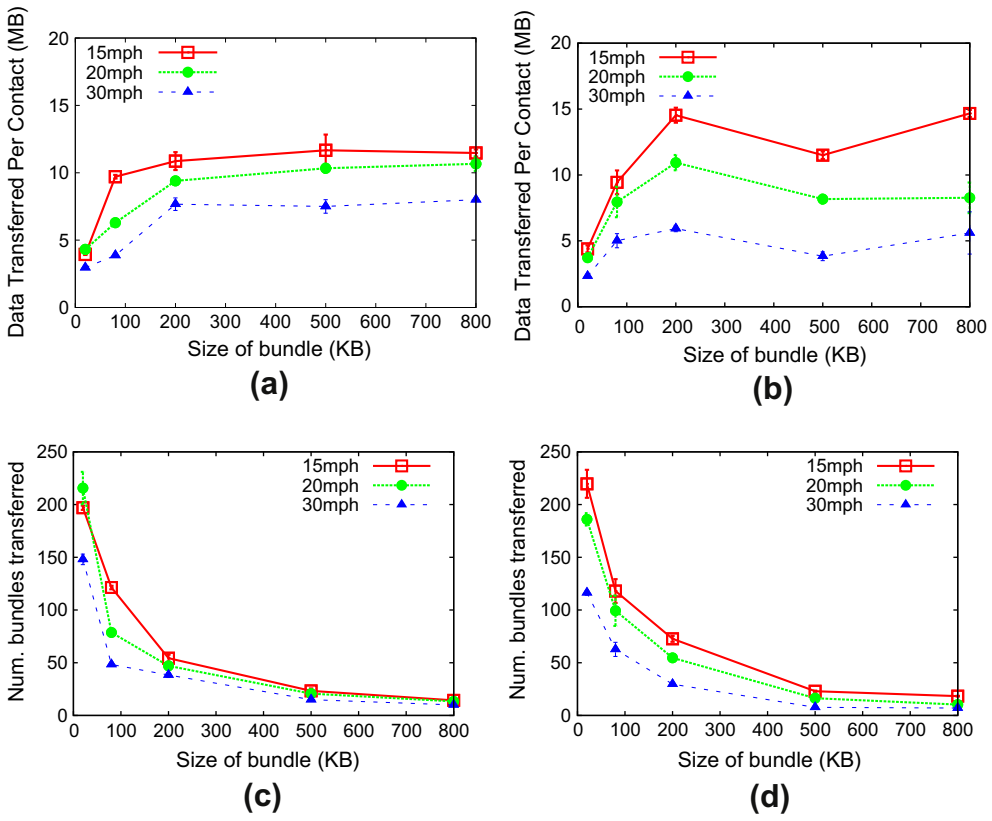


Fig. 13. Per-contact performance for WiFi: the DTC without security (a) and with security (b); the number of bundles transferred without security (c) and with security (d).

burning eyes and difficulty breathing per EMS/Fire dispatch. Unknown casualties. Engineer stated that train had radiological cargo; however the manifest has not yet been obtained. A full RECON by HAZMAT and aerial surveillance must be conducted in order to establish appropriate DECON. Winds are out of the north west. Initial HOT Zone has been established, see map. Any RECON by personnel shall be coordinated with HAZMAT. Responder medical and Command post located next to EOTC.

The area of the disaster was 0.081 square miles. No human movement was allowed in the HOT zone due to danger of contamination. Any robot entering the HOT or WARM zones had to be decontaminated at DECON before being handled by a human. The objective of our team was to provide networking in the area, so that data from the DECON area (DECON in Fig. 12) could reach the command post. The only travel route to the DECON area was through a dirt road (TRAVEL ROUTE in Fig. 12). Additionally, data from the marine robots (location "3" in Fig. 12) also needed a path to make it to the Command Post. Location 2 had a collapsed strip mall that the US&R team would have investigated. After a group huddle, we decided to deploy four routers in the area (locations 1, 2, 3, 4 in Fig. 12). Location 1 was a Data Waypoint, location 2 was a Sensor Proxy due to the presence of BTag Sensors nearby, and location 3 was a Data Waypoint. A Base Station (4) was placed at the EOC. The entire deployment effort took about two hours including travel time. Deploying a node was easy, since they were all mounted on tripods and were battery powered. BTag sensors were attached to the strip mall rubble using duct tape and were programmed using Smartphones/BTag app carried by our team. Additionally, a few pictures were sent using smartphones/file sharing app from the node at location 1 to the Base Station 4 at the EOC. FogNet service optimization using source routing was not performed due to the limited time available and the small size of the deployment area. Epidemic routing was used to route packets. The entire experiment, including setup and debugging took about six hours (excluding a lunch break).

6.2. WiFi contact benchmark

As mentioned before, DTC between a vehicle and a stationary node over DTN depends on many factors, two of them being the speed of the vehicle and the payload size. Other factors that can be considered include TCP over UDP, IPv6 over IPv4, the transmit power of the nodes and the distance of the vehicle from the stationary node. For reasons of simplicity and limited resources, only the speed and payload size were considered.

Data was gathered over multiple days involving around 50 man-hours. A node mounted on a tripod was placed at approximately the same location, within human error, on each day. A vehicle then drove by at three different speeds of 15, 20 and 30 mph. For each speed, several bundle sizes of 20 KB-800 KB were tested, with TCP over IPv4 over 802.11 IBSS mode on 5 GHz. For each combination of speed and bundle size, three runs were performed and averaged to iron out any random errors. The entire experiment was repeated for the case when DTN layer security is enabled. The results can be seen in Fig. 13.

[Effect of vehicle speed] The DTC for Wifi contacts is shown in Fig. 13(a,b). We observe that as speed increases, the DTC decreases, for both secure and insecure transfers. This is to be expected since increased mobility results in lesser contact time and also degradation of link quality. Considering a base speed of 15 mph, an increase of 1.3×/2× to 20/30 mph should intuitively result in a throughput decrease of 0.75×/0.5× respectively. Without security and when averaged over all bundle sizes, these ratios were in practice found to be 0.86× (+14.67%) and 0.63× (+26%). With security enabled, they were 0.72× (−4%) and 0.42× (−16%) – this is expected since security incurs overhead. The number of bundles transferred can be seen in Fig. 13(c,d). The ratios for insecure transfer were 0.91×/0.63×, whereas for secure transfers, it was 0.81×/0.49×. We conclude that the number of transferred bundles decreases sub-linearly with increase in speed.

[Effect of payload size] The payload size affects the number of unique bundles created in each node's queue. A large number of bundles stored on a router demands more resources for local processing. As a result, DTC is low at small bundle sizes. We also observe a flattening of the goodput curve at larger payload sizes (Fig. 13(a,b)), above around 200 KB. This implies that bundles with at least 200 KB are optimal for transferring between routers. Concretely, with a baseline of 20 KB, the expected ratios for 80, 200, 500, 800 are 4×/10×/25×/40×. Without security, these ratios when averaged over all speeds are: 1.77×/2.49×/2.63×/2.68×. With secure transfers, we achieve 2.15×/3×/2.25×/2.73×. These results lead us to believe, based on empirical data, that the DTC increases with the square of the expected ratio up to a "critical" bundle size, after which it remains constant. A similar effect holds true for the number of bundles transferred – the number stays almost constant once a threshold bundle size is reached.

6.3. 802.15.4 Contact benchmark

The effect of payload size and mobility on the DTC two nodes over 802.15.4 can be seen in Fig. 14. A stationary node was placed 5 feet above the ground, while a mobile node was placed in the passenger seat of a vehicle. The vehicle made multiple, regular runs at speeds of 15, 20 and 30 mph, transferring at each run multiple packets. These packets had a payload size between 10 and 90B. The software used was IPv6/UDP using Blip2 on TinyOS. The maximum MTU for 6lowpan/IPv6 is 100B – however, fragmentation was found to be unstable and hence unusable. Therefore, we limited our payload size to roughly 100B in our experiments. Each transferred packet used application layer acknowledgments. Each data point represents two separate runs using identical parameters.

[The effect of vehicle speed] upon DTC can be seen in Fig. 14(a). As expected, DTC decreases with increasing speed. DTC shows degradation consistent with speed: the ratios for 20/30 mph considering 15 mph as the baseline are 0.76×/0.55×, which are +1.3%/+10%. Packet losses (Fig. 14(b)) are independent of speed: the respective ratios are 0.99× and 1.01× when averaged across all payload sizes. We conclude that speed has a marked, linear effect on DTC, but packet loss percentages are independent of speed and constant.

[The effect of payload size] upon DTC is surprising: given the same contact time/speed, it increases with increasing payload size. This means that there is a constant overhead involved in the processing, sending and ACKing of packets – increasing the payload size does little to affect this overhead, but results in a large DTC. Given a baseline of 10B, the ratios for 30/50/70/90B (3×/5×/7×/9×) are 2.74×, 4.97×, 6.33×, 10.22×. The respective change against their expected ratios (assuming a linear relationship) are −8.65%, −0.61%, −9.51%, 13.54%. Thus, choosing the biggest possible payload size results in maximal DTC (within the limits of fragmentation). For packet loss (%), the ratios were 1.2×/1.15×/1.45×/0.9× – showing that

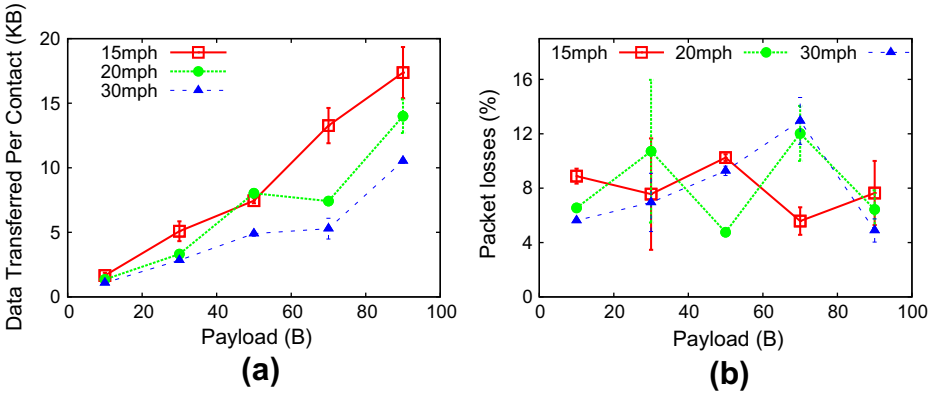


Fig. 14. Zigbee contact performance – (a) data transferred per contact and (b) packet loss per contact.

empirically, the largest payload size will suffer relatively fewer losses.

6.4. Sink election

The results of running a sink election algorithm using RPL/Blip2/TinyOS on EPIC motes is shown in Fig. 15. Four BTag Sensors were deployed outdoors, while a fifth node in a vehicle acted as a beacon with an interval of 2 s. The battery voltage was read every 2 s, the election was conducted every 10 s, and data was generated every 60 s. Because the experiment was conducted outdoors, we had to set the transmission power to −15 dBm (9.9 mA draw) to simulate a multi hop network. The vehicle drove in loops at around 10 mph first around nodes 1 and 2, then around nodes 3 and 4 starting at iteration 10. We see that the node with the highest score elected as the sink (Node 1 first, and then Node 4) – this verifies the correct execution of our algorithm. Interestingly, Node 3 was able to receive a few stray beacons due to radio irregularities. However, this did not disturb the election or cause a switch in the elected sink in the long run (up to iteration 45).

6.5. Networking

The DTN simulator chosen for the task of choosing a routing protocol was the Opportunistic Network Environment simulator (TheONE). The paths of vehicles were digitized using the Google Earth GIS software. The entire setup in simulation consists of 26 nodes – 5 data sources each at Sources 1 and 2, three routes with 5 vehicles on each route and a C2 node. The vehicles move at a speed uniformly chosen between $\mathcal{U}(19, 21)$ mph by default. The transmission range is 13 m, in order to allow for a multi hop network. Each data source sends a packet of size $\mathcal{U}(95, 105)$ KB every $\mathcal{U}(20, 30)$ seconds to the C2. The total simulation time for each scenario is 1 h unless specified otherwise.

The combined performance of four DTN routing protocols can be seen in Fig. 16. Epidemic routing aims to deliver messages by delivering a copy of the data to every neighbor that it encounters. It has one of the highest delivery rates and the lowest latency. A caveat is the overhead which is the number of extra messages created while routing per delivered packet, and the average hop count. In PRoPHET routing, each node maintains the probability of each of its neighbors being able to deliver a packet to a given destination. The delivery rate is same as that of Epidemic, but the latency is 26 s higher. SNW refers to the Spray and Wait protocol which aims to bound the number of copies of a packet in the network by a given parameter. For 2 values of 10 and 6, the SNW protocol has a very low overhead and hop count, but has a low delivery rate and high latency. We chose to use Epidemic routing for its high delivery rate and low latency.

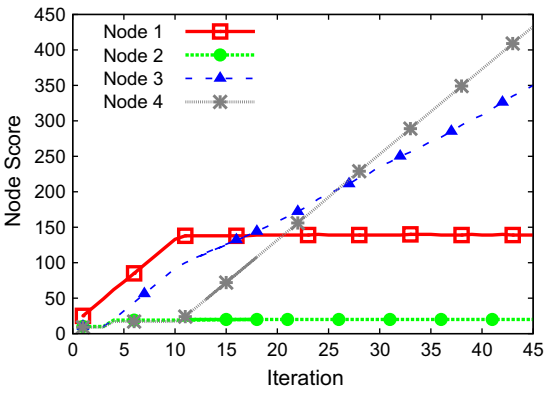


Fig. 15. Sink election evaluation.

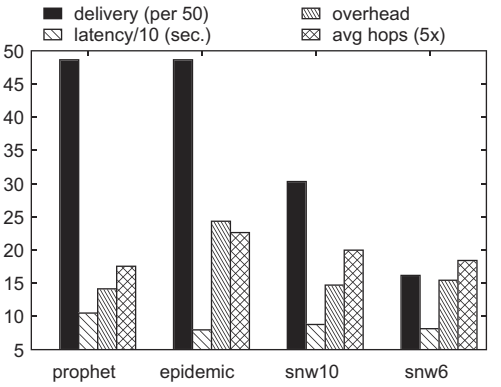


Fig. 16. Evaluation of latency, delivery rate, overhead and average hop count for four different DTN routing protocols in simulation.

6.6. Goodput and waypoint placement

[Setup] In order to evaluate the waypoint placement algorithm, we designed a deployment (Fig. 17) involving three cars and three flows with three data producing/consuming nodes. Flow1 in the following text refers to data sent from Source1 to the Base, Flow2 is from Source2 to

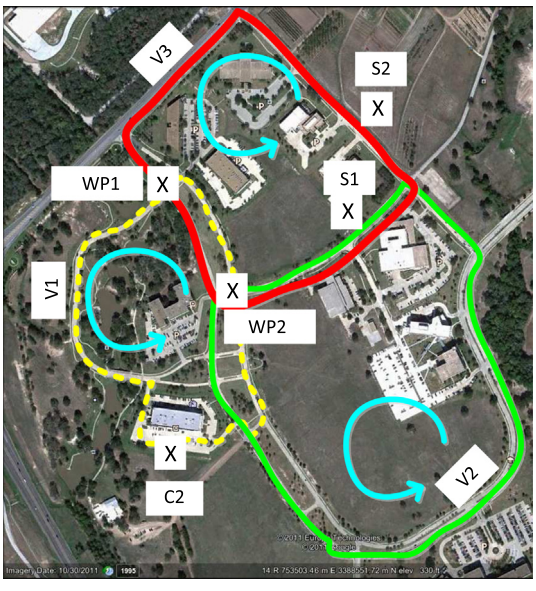


Fig. 17. Map of Deployment: S1, S2 are sources, V1, V2, V3 are vehicles, WP1, WP2 are Data Waypoints, X are locations.

the Base, and Flow3 is from Source1 to Source2. Flows1 and 2 were firm flows, whereas Flow3 was a potential flow from S2 to the Fog with a 33% availability. For the sake of flow diversity, Flow3 was made firm by choosing S2 as a destination. Two possible waypoint locations were WP1 and WP2. There are four configurations possible with two locations: none (config0), both (config3), WP1 only (config1) and WP2 only (config2). The goodput for each flow was experimentally measured for all possible waypoint configurations.

[Expt. 1: Epidemic routing/no DTC optimization] For this experiment all nodes performed epidemic routing. Results for the goodput and delay are presented in Fig. 18(a,b). Payload size was chosen to be 100 KB, each flow generated data at every 20 s, vehicles moved at around 20 mph. It has to be noted that the sources themselves act as data waypoints due to the nature of Epidemic routing – hence, the goodput improvement between the configurations is not that high as compared to the following experiment. Config3 proved to be optimal by providing the highest aggregate goodput across all the flows. If we consider only Flows1 and 2, since Flow3 does not need any additional waypoints (since the same vehicle passes through Sources1 and 2), configs1 and 3 are almost equal.

[Expt. 2: Source routing/with DTC optimization] For this experiment, values from the WiFi and 802.15.4 contact experiments were used to determine the payload sizes of flows. In addition to source routing replacing Epidemic routing of the previous experiment, security at the DTN layer was enabled. Flow2 was converted to a 802.15.4 flow with the vehicle picking up data from Source2 using 802.15.4 instead of Wifi. Flow1 was still Wifi based – this meant all deliveries to Source2 were made over Wifi. The payload sizes for Wifi and 802.15.4 were chosen to be 300 KB and 90B respectively. However, once a vehicle picked up 90B packets, they were marshaled into 300 KB DTN bundles. Data was generated every 30 s.

As a result we see a marked increase in the aggregate goodput as compared to the previous experiment. Because of source routing, unnecessary copies of bundles were not created, leading to efficient and non-redundant per-contact data transfer. However, the maximum delay in config0 is high because there is no data replication (and only opportunistic contact between vehicles), but when waypoints are present, the delay is comparable in spite of the increased payload size and overhead due to security. We conclude that using source routing and choosing the payload size optimally results in a 2× increase in goodput.

6.7. Apps: Vibration monitoring

We evaluated our building sensing network in Disaster City on three different rubble piles: one consisting of wooden rubble (Fig. 19(a)), one of concrete, and another with a combination of concrete and mud. In the latter one, the soft mud dampens the vibrations caused inside the pile and hence makes detection with a seismic sensor difficult. Samples for different types of events were gathered at each

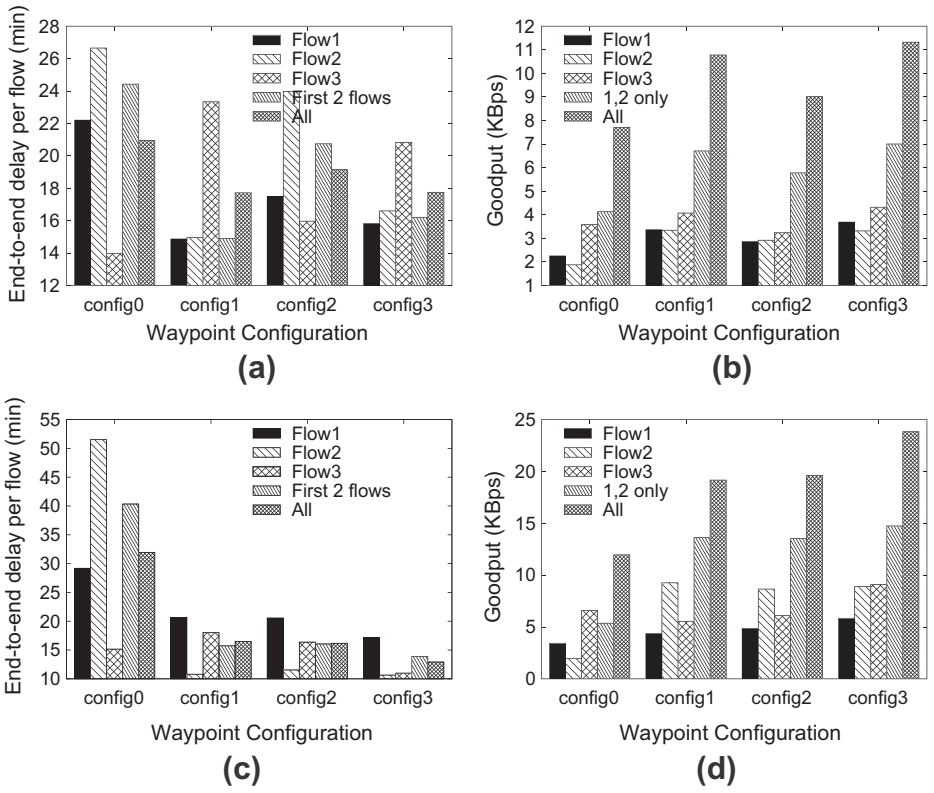


Fig. 18. Expt. 1: DWP performance in terms of (a) latency and (b) goodput; Expt. 2: DWP performance in terms of (c) latency and (d) goodput.

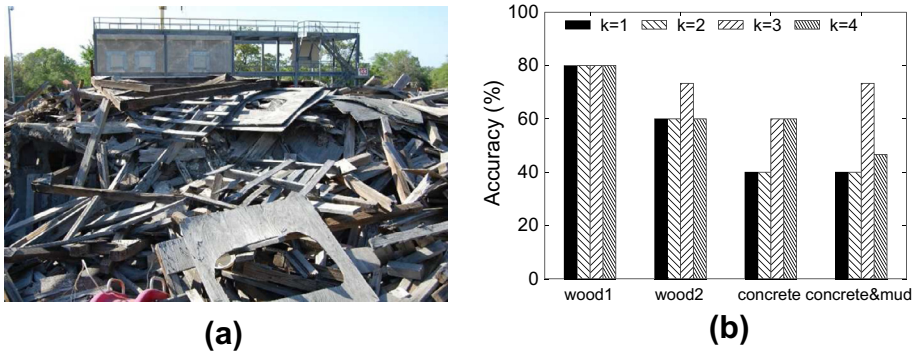


Fig. 19. (a) Wooden rubble pile in Disaster City and (b) KNN classifier accuracy as a function of k.

of these piles: a stone drop, a footstep and a hammer strike. Half of the samples were used to train the KNN classifier, and the other half to evaluate performance. All samples were taken at slightly different strike intensities and distances from the sensor.

Results are shown in Fig. 19(b). "wood1" represents samples taken at the wooden pile with the default sensitivity threshold of 25 and "wood2", at a threshold of 50. A higher threshold implies lower sensitivity. This higher threshold was not possible on the two other piles since the sensor could not register soft knocks and events. We conclude that a $k = 3$ provides for optimal performance from the KNN classifier with an average detection of accuracy of 73.33% independent of type of rubble, strike intensity and the distance from the seismic sensor.

6.8. Apps: Separation detection

The effect of separation upon the state of a team member is shown in Fig. 20. An experiment was conducted inside an urban building where iPod touches running the separation detection app were given to each member. "One" is the team leader and hence injects a constant state into the network. Initially, all the team members were present in a single room until time 30. Then, One and Two separated from Three by going into another room.

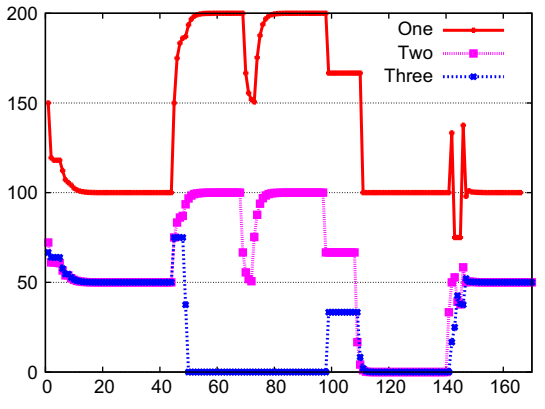


Fig. 20. Graph of state versus time for a team of three responders.

As a result, the state of Three drops to zero since it is no longer connected to One, and the states of Two as well as One increase and converge (time 40 – 60).

Then, One and Two move around in the large room with lot of metallic wall sized objects, causing disconnection. This disconnection is temporary and does not signal a separation. Later, Two returns to the same room as Three at time 95. As a result, the state of Three increases for time 100 – 110 due to the residual state brought by Two, but both of them quickly decrease to zero at 110 since they are no longer connected to One. Finally, One reunites with Two and Three at 140 causing all of their states to converge once again to their initial values. The average detection delay, looking at each of the three separation events and the corresponding state at that time, is $\frac{(45-30)+(98-95)+(143-140)}{3} = 7$ s. The detection delay for separation as opposed to rejoining is a little longer because of the guard interval before a node declares a neighbor as disconnected.

6.9. Lessons learned

In this section, we share some of the valuable lessons we learned during our experience of designing, building and testing DistressNet over a year and about 1500 man-hours of outdoor deployments.

Power savings: In order to truly extend the lifetime of the system, duty cycling the 802.11 radios which draw around 200 mA when active is necessary. Because the majority of the devices in DistressNet operate in the IBSS mode, power saving mode (PSM) support for IBSS mode in the hardware is essential. However, implementing this functionality in the linux ath9k drivers was too time consuming. Hence, we could only enable PSM functionality for class B STAs while the 2.4 GHz interface of class C devices operated in AP mode. Experimental verification was performed using MiniPCI extenders which allowed us to isolate and measure the current drawn by a MiniPCI card, used in the RB433UAH as well as older laptops.

Linksys WRT54GL routers meant for home-office use exhibit duty cycling when turned on and off using the *iwconfig* tool, when flashed with OpenWRT. This reduces the current drawn by 70 mA, while not allowing the established IP layer connections to time out. While this behavior

does save some power, this is not a clean solution which is exhibited by all devices. An implementation in the kernel provides much better results and was pursued by us in favor of this approach.

Multiple hardware generations: Initial versions of DistressNet attempted to use Linksys WRT54GL as well as Netgear WNDR3700 routers. "Hardened" routerboards such as the Mikrotik RB433UAH were eventually chosen due to the robustness as well as the customizability. Due to the increased disk space available from 8 MB to 512 MB in RB433UAH, we were able to include the libraries needed for DTN security, without compromising or trying to optimize the size of other libraries. This approach while slightly more expensive allowed for much more luxurious debugging and re-use of popular libraries like Boost for C++. The USB interfaces as well as dual radios on the WNDR3700/RB433UAH provided additional functionality while at the same time saving the unreliability that comes from runing multiple daemons on the same radio interface.

7. Conclusions

We have presented DistressNet, a system that addresses the needs of Urban Search and Rescue workers during the aftermath of a natural disaster. An app oriented design integrates inexpensive and heterogeneous battery-powered COTS devices like smartphones and low power sensors into an easily deployable architecture. External cloud services such as Amazon S3 and Twitter are made available to first responders over a delay tolerant network, through the FogNet subsystem. Situational awareness is improved by increasing the aggregate throughput of DistressNet, by optimally by placing additional hardware at certain locations in the area of deployment. The vibration sensing app can distinguish victims trapped under a rubble pile from environmental noise with 73.3% accuracy.

DistressNet is an academic effort and has been evaluated in realistic settings wherever possible. Evaluation of the system as a whole, by scaling DistressNet to tens of nodes and hundreds of flows over multiple days is time and resource consuming – therefore, we have only performed piecewise evaluation. The waypoint placement algorithm assumes that mobility in the area is a subset of the PDM model, and will possibly benefit from a more comprehensive model.

Our agenda for ongoing and future work on DistressNet include thorough evaluation over multiple days. Extensibility of the DistressNet architecture allows for exciting applications such as victim triangulation under rubble using seismic sensors to be easily integrated. System-wide energy efficiency has not been sufficiently addressed, and is ongoing work – as is the problem of optimizing metrics other than the throughput, like mean delay or packet delivery rate.

Acknowledgements

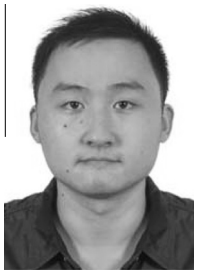
This work was funded in part by NSF Grants #1127449, #1145858, #0923203.

References

[1] http://www.bbc. co.uk/news/world-asia-pacific-12709850.
[2] http://en.wikipedia.org/wiki/2010 _Haiti_earthquake.
[3] http://www.fema.gov.
[4] Army to deploy iphones in combat. <http://goo.gl/ZKzdY>.
[5] H. Chenji, A. Hassanzadeh, M. Won, Y. Li, W. Zhang, X. Yang, R. Stoleru, G. Zhou, A wireless sensor, adhoc and delay tolerant network system for disaster response, Tech. Report 2011-9-2, Texas A&M, 2011.
[6] US&R Task Force Equipment. <http://www.fema.gov/emergency/usr/equipment.shtm>.
[7] http://www.project25.org/.
[8] Project 25 interoperable communications for public safety agencies. http://www.motorola.com/web/Business/Solutions/Business Solutions/Mission Critical Communications/ASTRO 25 Trunked Solutions/_Document/Project 25 Whitepaper.pdf.
[9] O. Chipara, W.G. Griswold, A.N. Plymoth, R. Huang, F. Liu, P. Johansson, R. Rao, T. Chan, C. Buono, WIISARD: a measurement study of network properties and protocol reliability during an emergency response, in: MobiSys, 2012.
[10] S.W. Brown, M. William, G. Griswold, B. Demchak, B. Leslie, A. Lenert, Middleware for reliable mobile medical workflow support in disaster settings, in: AMIA Annu. Symp. Proc., 2006.
[11] M. Arisoylu, R. Mishra, R. Rao, L.A. Lenert, 802.11 Wireless infrastructure to enhance medical response to disasters, in: AMIA Annu. Symp. Proc., 2005.
[12] L.A. Lenert, D.A. Palmer, T.C. Chan, R. Rao, An intelligent 802.11 triage tag for medical response to disasters, in: AMIA Annu. Symp. Proc., 2005.
[13] J.P. Killeen, T.C. Chan, C. Buono, W.G. Griswold, L.A. Lenert, A wireless first responder handheld device for rapid triage, patient assessment and documentation during mass casualty incidents, in: AMIA Annu. Symp. Proc., 2006.
[14] F. Peña-Mora, A. Chen, Z. Aziz, L. Soibelman, L. Liu, K. El-Rayes, C. Arboleda, T. Lantz, A. Plans, S. Lakhera, S. Mathur, Mobile ad hoc network-enabled collaboration framework supporting civil engineering emergency response operations, Journal of Computing in Civil Engineering 24 (3) (2010).
[15] K. Fall, G. Iannaccone, J. Kannan, F. Silveira, N. Taft, A disruption-tolerant architecture for secure and efficient disaster response communications, in: ISCRAM, 2010.
[16] H. Soroush, N. Banerjee, A. Balasubramanian, M.D. Corner, B.N. Levine, B. Lynn, DOME: a diverse outdoor mobile testbed, in: HotPlanet, 2009.
[17] X. Tie, A. Venkataramani, A. Balasubramanian, R3: robust replication routing in wireless networks with diverse connectivity characteristics, in: Mobicom, 2011.
[18] S. Mehrotra, C. Butts, D. Kalashnikov, N. Venkatasubramanian, R. Rao, G. Chockalingam, R. Eguchi, B. Adams, C. Huyck, Project rescue: Challenges in responding to the unexpected, in: SEISC, 2004.
[19] R.B. Dilmaghani, R.R. Rao, An ad hoc network infrastructure: communication and information sharing for emergency response, IEEE ICWMCNC (2008).
[20] B. Manoj, A.H. Baker, Communication challenges in emergency response, Communications of the ACM 50 (2007).
[21] R.B. Dilmaghani, B.S. Manoj, B. Jafarian, R.R. Rao, Performance evaluation of rescue mesh: a metro-scale hybrid wireless network, in: WiMesh, SECON, 2005.
[22] B. Xing, S. Mehrotra, N. Venkatasubramanian, RADcast: enabling reliability guarantees for content dissemination in ad hoc networks, in: INFOCOM, 2009.
[23] Network-centric warfare and wireless communications. <http://goo.gl/xYZtw>.
[24] W. Zhao, Y. Chen, M. Ammar, M. Corner, B. Levine, E. Zegura, Capacity enhancement using throwboxes in dtns, in: MASS, 2006.
[25] N. Banerjee, M.D. Corner, D. Towsley, B.N. Levine, Relays, base stations, and meshes: enhancing mobile networks with infrastructure, in: Mobicom, 2008.
[26] N. Banerjee, M. Corner, B. Levine, An energy-efficient architecture for dtn throwboxes, in: INFOCOM, 2007.
[27] M. Ibrahim, P. Nain, I. Carreras, Analysis of relay protocols for throwbox-equipped dtns, in: WiOPT, 2009.
[28] M. Farukh Munir, A. Kherani, F. Filali, Stability and delay analysis for multi-hop single-sink wireless sensor networks, in: PerCom, 2008.
[29] http://sema.dps.mo.gov/newspubs/SRTemplate.asp?ID=N09110049.
[30] http://www.fema.gov/emergency/usr/equipment.shtm.
[31] http://www.designnews.com/document.asp?doc_id=213077.

[32] A.L. Hughes, L. Palen, Twitter adoption and use in crisis twitter adoption and use in mass convergence and emergency events, in: Proceeding of the 6th International ISCRAM Conference, 2009.
[33] J. Sutton, L. Palen, I. Shklovski, Backchannels on the front lines: emergent uses of social media in the 2007 Southern California wildfires, in: ISCRAM, Washington, DC, 2008.
[34] A. Bahill, R. Botta, Fundamental principles of good system design, EMJ – Engineering Management Journal 20 (4) (2008) 9–17.
[35] G. Guven, E. Ergen, Identification of local information items needed during search and rescue following an earthquake, Disaster Prevention and Management 20 (5) (2011).
[36] P. Barooah, H. Chenji, R. Stoleru, T. Kalmar-Nagy, Cut detection in wireless sensor networks, IEEE TPDS 23 (2012) 483–490. March.
[37] M. Uddin, D. Nicol, T. Abdelzaher, R. Kravets, A post-disaster mobility model for delay tolerant networking, in: Winter Simulation Conference (WSC), 2009.

Harsha Chenji (cjh@cse.tamu.edu) joined the Department of Computer Science and Engineering at Texas A&M University in August 2007. He is currently a Ph.D. candidate under the guidance of Dr. Radu Stoleru, after graduating with a M.S. (Computer Engineering) degree in December 2009. He obtained his Bachelor of Technology in Electrical and Electronics Engineering from the National Institute of Technology Karnataka, Surathkal, India in May 2007.



Wei Zhang (wzhang@cse.tamu.edu) is currently pursuing his Ph.D. in Computer Engineering under the guidance of Dr. Stoleru since 2011. He completed his B.S and M.S in Electrical Engineering from Northwest Polytechnical University Xi'an in 2005 and 2008 respectively.

Radu Stoleru (stoleru@cse.tamu.edu) is currently an assistant professor in the Department of Computer Science and Engineering at Texas A&M University. He is the recipient of the prestigious NSF CAREER award in 2013. He received his Ph.D. in computer science from the University of Virginia in 2007, and the Computer Science Outstanding Graduate Student Research Award in 2007. His research interests are in deeply embedded wireless sensor systems, distributed systems, embedded computing, and computer networking. He has authored or co-authored over 70 conference and journal papers with over 2200 citations.

Clint Arnett (clint.arnett@teexmail.tamu.edu) is a Project Manager for the Texas Engineering Extension Service (TEEX), Urban Search and Rescue (US&R) Division. He has over seven years experience in the fields of search and rescue (SAR) and disaster response. Additionally, he works in the TEEX ESF-9 Coordination Center, and assists the Texas Task Force 1 Logistics Section both in training and deployments. Clint also has 17 years of experience in all phases of the construction equipment industry; encompassing management, sales, maintenance and repair, and transportation, as well as additional experience in hazardous materials transportation.