Achieving Reliable Communication in Dynamic Emergency Responses

Octav Chipara, Anders N. Plymoth, Fang Liu, Ricky Huang, Brian Evans, Per Johansson, Ramesh Rao, William G. Griswold
University of California, San Diego

Abstract

Emergency responses require the coordination of first responders to assess the condition of victims, stabilize their condition, and transport them to hospitals based on the severity of their injuries. WIISARD is a system designed to facilitate the collection of medical information and its reliable dissemination during emergency responses. A key challenge in WIISARD is to deliver data with high reliability as first responders move and operate in a dynamic radio environment fraught with frequent network disconnections. The initial WIISARD system employed a client-server architecture and an ad-hoc routing protocol was used to exchange data. The system had low reliability when deployed during emergency drills. In this paper, we identify the underlying causes of unreliability and propose a novel peer-to-peer architecture that in combination with a gossip-based communication protocol achieves high reliability. Empirical studies show that compared to the initial WIISARD system, the redesigned system improves reliability by as much as 37% while reducing the number of transmitted packets by 23%.

Introduction

Emergency responses are unplanned, highly dynamic, and vary in size from the involvement of a few first responders to hundreds. A typical emergency response involves triaging victims, providing minimal care to stabilize their condition, and transporting them to hospitals based on the severity of their injuries. As the scale of a disaster increases, the time victims spend on scene also increases allowing for re-triage and further, more comprehensive, medical treatment. Traditionally, victims have been identified and their medical condition tracked using pen and paper. Additionally, first responder efforts are usually coordinated verbally over noisy hand radios. These methods of gathering and sharing information are often prone to inaccuracies and may not scale well with the size of disasters [1].

Embedded devices and wireless technology hold the promise of transforming emergency responses. Persisting information in digital form has two key advantages: (1) the likelihood of information loss during chaotic emergency responses is reduced and (2) digital information is easier to be shared among emergency response personnel. This makes it feasible for incident commanders to have an improved situational awareness by assessing the progress of an emergency response in real-time. In addition, patient information can also be shared with the hospitals where victims will be treated to allow hospital administrators to properly allocate their limited resources. Yet, the use of technology is fraught with challenges that must be addressed in order to ensure the adoption of such a system.

Existing emergency response systems use either cellular [18,3] or mesh networks [12,13,2,14,15,16,17,4,5,6] to support communication among first responders. Cellular networks are used increasingly to deliver patient information as they are transported to the hospital. However, cellular networks may not be well suited for an emergency response system: major disasters such as earthquakes or hurricanes may damage or overload the cellular network potentially hindering its ability to relay information reliably. An alternative is to use mesh networking technology. In this case, first responders would carry a limited number of wireless nodes that, when deployed, would self-organize in a mesh network that facilitates communication among first responders at the disaster scene. In WIISARD, we adopt this latter approach to develop a self-contained emergency response system.

A major challenge in emergency response systems that use mesh networks is to develop networking protocols that disseminate information reliably to all first responders. This is not an easy task in a *dynamic wireless environment*! First responders will be *mobile* as they provide first aid to victims. Fire trucks, heavy equipment, and people attenuate radio signals to various degrees as they move within the environment and, as a result, link quality is subject to significant temporal variations. Moreover, *interference* from other wireless networks (e.g., video broadcast) can corrupt packets transmitted within the wireless network. Many wireless mesh networks that must support mobile entities are typically deployed in advance to ensure sufficient coverage. However, on a disaster scene, networking resources arrive

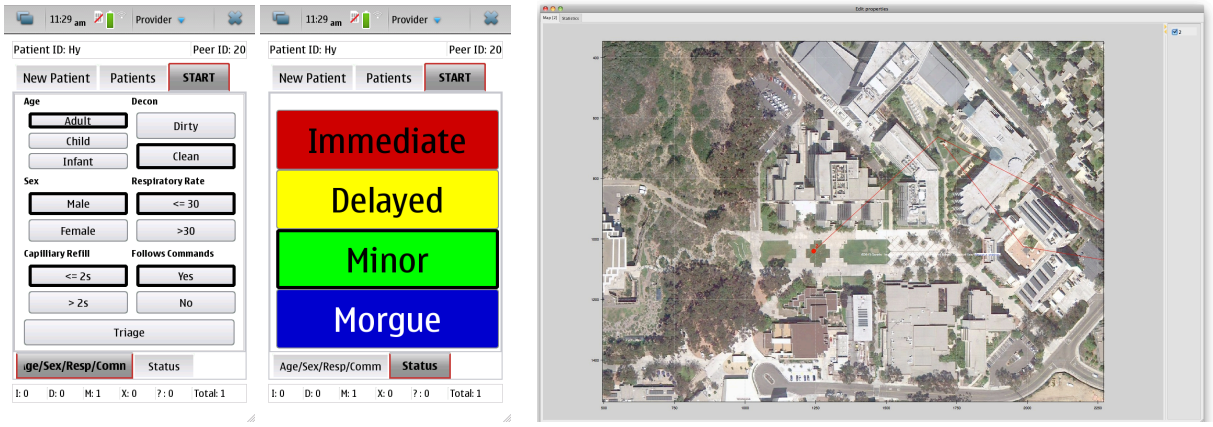
incrementally and it is unlikely that complete coverage will exist. Therefore, the system must achieve high reliability even in the presence of *frequent network disconnections* and *network partitions* due to lack of coverage.

The initial WIISARD protocol adopted a client-server architecture. An AODV-based ad hoc routing protocol was employed to maintain multi-hop routes between clients and the server. This design is not unique; other systems use a client-server architecture[12,13,2,14,15,16,17] or rely on multi-hop routing[2,4,5,6]. In this system architecture, the information collected by first responders is first transmitted to a server usually over multiple hops. The WIISARD server would cache all information and disseminate to clients any missing information. To handle disconnections, the clients cache data locally and attempt to contact the server periodically. A major drawback of this architecture is that if the network becomes partitioned important information may never reach the server. Observations from several disaster drills where WIISARD was deployed indicate that network portioning is likely to occur during emergency responses. We deployed the initial WIISARD system during multiple disaster drills, but due to conditions described above, results were mixed. In fact, in the Golden Guardian drill, less than 10% of the collected medical data was delivered by the completion of the drill.

This paper makes the following contributions: (1) We deploy the initial WIISARD system in an indoor 802.11 testbed to reproduce and understand the problems encountered during disaster drills. Our analysis indicates that three key factors contributed to poor reliability: existing ad hoc routing protocols are unable to maintain routes in the presence of mobility, in a client-server architecture the server becomes a bottleneck hindering the timely delivery of data, and frequent network disconnections prevent the dissemination of data. Since the performance issues are partly due to the client-server architecture, we expect that other client-server systems will suffer from similar reliability problems. (2) Based on these insights we propose a new network architecture that adopts a peer-to-peer architecture in which nodes act as both clients and servers. To ensure reliable communication in highly dynamic environments, we forgo the construction of multi-hop routes and present a new protocol, WIISARD Communication Protocol (WCP), in which each node exchanges information only with its one-hop neighbors. Each node provides its neighbors their missing information ensuring that all nodes eventually receive all data. (3) Empirical results show that, in contrast to the initial WIISARD system, the combination of a peer-to-peer architecture and WCP can achieve high reliability even in the presence of mobility and network disconnections.

System Description

Disaster response is about moving victims out of danger, tagging them, triaging them, treating them, and transporting them to hospitals as soon as possible. Additionally, real-time information about the progress of the response must be provided to incident commanders who are tasked with coordinating the emergency response. WIISARD adopts a multitude of technologies to support these activities effectively as detailed in the following:



(a) START Triage

(b) Command Center

Figure 1: START Triage and Command Center components

Tagging: The first task of first responders upon their arrival on the scene is to tag victims. Victims in WIISARD are tagged using passive RFIDs. The RFID identifiers serve as unique identifiers for each victim for the duration of the emergency response. Our original system included a barcode solution, however, we abandoned this approach due to the inconsistent performance of barcode readers under all light conditions (e.g., sunny days).

Triage: First responder devices combine a Nokia N900 mobile phone and an RFID reader. The phone and the reader communicate over Bluetooth. First responders begin the triage of a victim by scanning their RFID tag. A triage application that implements the Simple TriAge Rapid Treatment (START) protocol[7] runs on the mobile phone (see Figure 1(a)). The START triage application allows first responders to triage victims within a minute requiring minimal physiological information about the victim. A victim may be re-triaged as necessary using the same process.

Tracking: The Nokia N900 has GPS capabilities. WIISARD takes advantage of these capabilities to track not only the location of the first responders that carry the mobile phones but also the location of the victims. The location of victims is inferred based on the location of the providers: when a provider scans a victim's RFID, the victim's location is updated based on the provider's current position. Traditional emergency response systems do not provide such tracking capabilities.

Transport: A transport officer is responsible for selecting the hospital to which each patient is sent. To facilitate this key function, WIISARD implements a role-tailored user interface that allows the transport officer to send patients to hospitals according to their status as set by the other first responders. The transport application is deployed on tablet class devices to take advantage of the larger screen size of such devices.

Command Center: The command center is designed to present the incident commander with an overview of the situation (see Figure 1(b)). The command center is capable of accessing all patient information, create summaries about the progress of the response, and display the GPS coordinates of first responders and victims. The command center is also deployed on a tablet class device.

The remainder of this section focuses on the two communication architectures that we developed to support reliable and timely communication within WIISARD.

Initial WIISARD: *Client-Server Architecture and Ad-hoc Routing*

The initial implementation of WIISARD adopted a client-server architecture. The key advantage of this architecture is its conceptual simplicity. Each client (either mobile phones or tablets) transmits the information input by a first responder to the server. The server caches all information it receives. Clients provide a summary of the information each has stored locally to server periodically. In response, the server would transmit the missing information to each client. A standard ad-hoc routing protocol such as AODV may be used to route packets between clients and the server over multiple hops. Application-level retransmissions are employed in the exchanges between clients and server to ensure reliability in the face of variable link quality.

We deployed the initial WIISARD system as part of drills with mixed results. In the following, we will discuss some of the limitations of the initial WIISARD system, which we hypothesize to have contributed to the observed problems. Later, we will reproduce the issues encountered in the field on an indoor wireless testbed and analyze their impact on system reliability.

The initial WIISARD system suffered from three key limitations. A client-server architecture requires clients to have connectivity to the server in order to exchange information. This can pose problems in emergency response systems because, due to insufficient coverage, a client may have intermittent connectivity to the server. The initial WIISARD system alleviated this concern by caching all information generated by the client, and periodically (every 15 seconds) attempting to relay it to the server. While this mechanism is effective in resolving short-term disconnections, our experience with drill exercises shows us that clients may lose connectivity to the server for several minutes significantly hindering the emergency response: without connectivity to the server, the first responders cannot exchange information even when they are within the communication range of one-another.

In a client-server architecture, novel data from a client must be relayed to the server before the server transmits it to the remaining clients. This creates a bottleneck on the server that must process a significant number of packet requests.

As a result, the system is susceptible to prolonged delays and poor reliability when the server is overloaded.

The initial WIISARD system employed an AODV-based[8] ad hoc routing algorithm. A number of prior works found ad hoc routing protocols to provide low reliability in the presence of mobility[9,10]. Fundamentally, routing protocols require nodes to exchange multi-hop information about their connectivity to compute paths between end-points. In dynamic environments, where this information is subject to change, the network may spend significant resources updating routes without delivering the user data.

WIISARD: *Peer-to-Peer Architecture and Gossip*

In response to the problems observed during drills, we have redesigned WIISARD's network architecture and communication protocol. We adopted a peer-to-peer architecture in which a peer acts as both client and server. This has the advantage of allowing nodes within communication range to communicate without requiring connectivity to the server. Moreover, because the performance of ad hoc routing protocols may be affected by mobility and network dynamics, we dispense with the construction of multi-hop routes and opt for a gossip-based solution. A gossip protocol works by having each node "gossip" the information it hears from its neighbors until all nodes in the network have this information. Gossip protocols have the inherent advantage of being local protocols in which each node communicates only with the nodes within its communication range. However, gossip protocols suffer from the broadcast-storm problem: a node requests a piece of information and all nodes within the communication range rush to send it resulting in packet collisions and high overhead. Therefore, a key challenge that we must address is to disseminate information with minimum overhead.

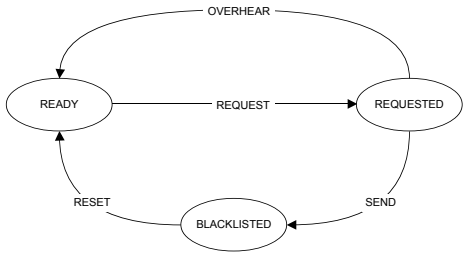


Figure 2: State diagram of a block in WCP

To address this challenge, we developed WIISARD Communication Protocol (WCP), a novel gossip protocol that disseminates data with low overhead. Data generated by peers is stored as `Blocks`. Each block has a globally unique key that includes the identifier of the peer that created the block and a sequence number. The bulk of the state maintained by WCP includes the blocks each peer receives. WCP does not maintain all blocks in memory, but rather saves them to disk to conserve memory. To minimize the I/O overhead, a simple caching scheme that maintains the most frequently used blocks in memory is used.

In WCP, a peer N succinctly describes its state (i.e., the blocks that N already received) in `Beacons`. A peer N considers the sequence of blocks created by each peer M. For each peer M, N includes two pieces of information in a `Beacon`. First, the sequence number of the first missing block in the sequence of blocks created by M is included. Second, a bit vector is included to indicate which blocks following the first missing block have already been received by N. As an example, the sequence 1, 2, 3, 6, 7, will be summarized to have the first missing block 4 and a bit vector 011 to indicate that block 5 is missing while 6 and 7 have been already received. The bit-vector encoding allows WCP to avoid retransmitting blocks that N has already received.

`Beacons` are transmitted periodically (once every 5 seconds). A peer R that receives a beacon from N can determine the `Blocks` that N misses. Peer R divides the current beacon period in s equal-sized `SendWindows`. Upon receiving a beacon, R creates a `Flow(N)` data structure that contains the list of s blocks that N is missing. In each `SendWindow`, peer R will transmit a `Block` from `Flow(N)` until the end of the beacon period. At the end of the

beacon period, the queue of `Flow(N)` will be empty.

It is important to note that unlike traditional routing protocols that maintain multi-hop path costs, WCP maintain minimal flow state. WCP assumes that the connectivity remain stable for the duration of a beacon period. If this is not the case, WCP will perform up to s transmissions that may be lost. By configuring the size of `SendWindow` and beacon period, we can effectively trade-off between the overhead of beacons and the period of time WCP assumes the presence of connectivity between nodes.

A concern in WCP is that multiple nodes may request the same `Block` within a short period of time. A single transmission of a `Block` may fulfill all outstanding requests due to the broadcast nature of the wireless medium. WCP handles this issue by associating with each `Block` some state information (see Figure 2). Each `Block` starts in state `READY`. Upon receiving a request for a `Block`, its state is changed from `READY` to `REQUESTED`. WCP will send a `Block` only if its state is `REQUESTED`. After transmitting a `Block`, WCP changes its state to `BLACKLISTED`. This effectively prevents the same `Block` to be retransmitted again when multiple peers request it within a short period of time. The state of a `Block` is reset to `READY` after a time out equal to the beacon period. This allows subsequent requests for the same block to be fulfilled when there are peers that still require it.

Due to the broadcast nature of wireless, multiple peers usually receive a beacon. Their responses must be coordinated to reduce the likelihood of receiving the same blocks from different peers. To suppress redundant transmissions WCP relies on overhearing and randomized transmissions. When WCP overhears a `Block` it will check if the block is requested. If the state of the block is `REQUESTED`, the request is cancelled and the state is set to `READY`. In addition, the order in which blocks are transmitted is randomized over the `SendWindows`. Note that in the case when fewer blocks than the number of `SendWindows` are requested, some of the windows will remain unused. The randomized mechanism ensures that it is unlikely for two peers to transmit the same `Block` in the same `SendWindow`. Overhearing a `Block` will cancel requests, effectively reducing the number of redundant transmissions.

There are situations when the cancellation mechanism may actually prolong the time to disseminate information. Consider a scenario in which nodes A, B, C, and D communicate over links (AB), (BC), and (CD). In this scenario, nodes B and C have a block Xthat is requested by both nodes A and D. Depending on the interleaving of requests and responses, it is possible that A's request will go unanswered. This occurs when node B overhears node C transmitting block X (in response to D's request) and cancels its response to A. Two points are worth noting. First, node A will eventually receive the missing block X, since it will request it again according to the protocol. Second, the cancellation mechanism is most effective in dense networks. In this case, the decision to cancel requests significantly reduces overhead since a single reply fulfills may requests. Given the significant range of WiFi cards, we expect that this will be the common case.

Empirical Study

The empirical study is designed to reproduce some of the challenges observed during drills with the initial WIISARD system. The goal of the study is to compare the reliability of two network architectures. The experiments were conducted on a wireless testbed consisting of 11 mesh boxes that covers the 6-th floor of Atkinson Hall at University of California, San Diego (see Figure 3). Mesh boxes are equipped with two IEEE 802.11 b/g cards and run Ubuntu Linux. During the experiments one of the wireless cards is configured in promiscuous mode to capture transmitted packets. Mesh boxes are connected to the local LAN to allow for sharing of the collected data and time synchronization via NTP. The logs from the mesh boxes were time synchronized and merged into a single log that was used for analysis. In addition, Nokia 900 mobile phones with IEEE 802.11 b/g cards were used for the mobile users. In the client-server architecture, we used AODV-UU[11] from Uppsala University to route packets. AODV-UU is an implementation of the AODV RFC 3561[8] and has been widely used by the ad hoc networking community as a baseline.

Disconnected Scenario

The first experiment considers a scenario where there are prolonged network disconnections. As previously discussed, this scenario occurs frequently during emergency responses when there are insufficient network resources to provide coverage at the disaster scene. Node 61 was selected as the server and three additional mobile phones were used. Two of phones were located at positions A and B as indicated in Figure 3. The other mobile phone is carried by a mobile

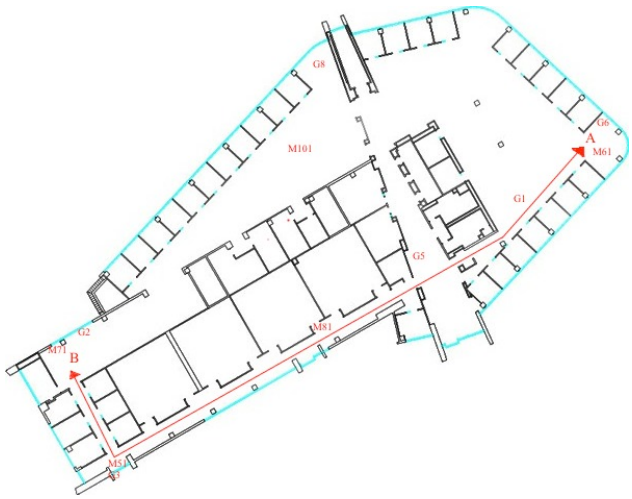
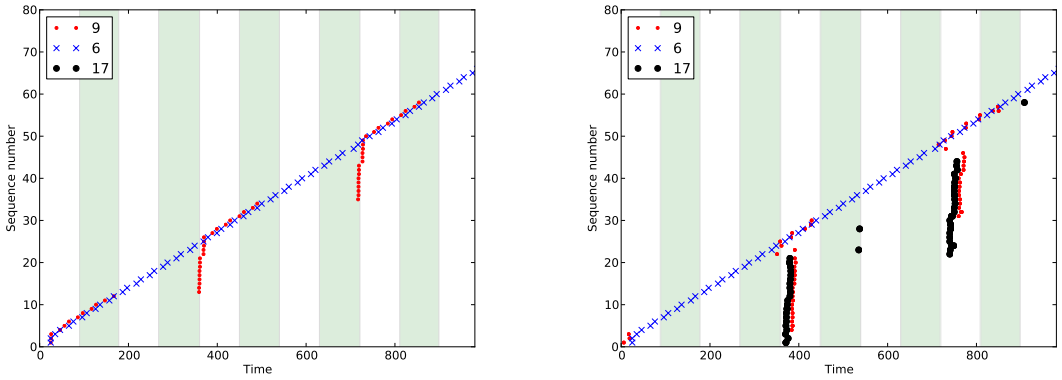


Figure 3: UCSD-CalIT2 Wireless Testbed

user which moves according to the following motion pattern: the user starts at location A where it remains for 1:30 minutes, then the user moves from A to B where it remains for 1:30 minutes before returning back to A. This motion was repeated 4 times. The phones generate a block every 15 seconds.



(a) Client-Server + AODV

(b) Peer-to-Peer + WCP

Figure 4: *Impact of disconnections:* For the client-server architecture the data generated on node 17 never reached the server. In contrast, in a peer-to-peer architecture data from node 17 is disseminated through data muling.

During the experiment we recorded the time when each block created by the mobile nodes was received by node 61. Based on this information, in Figure 4, we plot the sequence number and reception time at node 61 of each block created by a mobile phone. The shaded time intervals indicate the periods during which user was mobile.

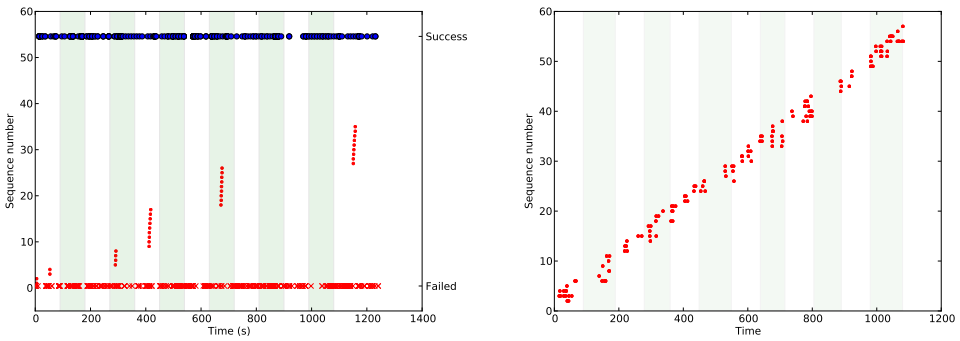
Figure 4(a) shows the performance of the client server architecture when AODV is used for routing. For the duration of the experiment, node 6 was located at position A which is close to node 61. As a result, node 6 had a reliable path to the server. The blocks created by node 6 were delivered timely as indicated by the absence of gaps in its trace. In contrast, node 9 was carried by the user and had intermittent connectivity to the server. This is visible in the large gaps that indicate that blocks were buffered until a path to the server was established. For the duration of the experiment, no data from node 17 was delivered to the server. This occurred even though an intermittent path between the server and node 17 existed. However, AODV was unable to take advantage of it due to its short life.

Figure 4(b) plots the results obtained when the peer-to-peer architecture in conjunction with WCP was used. In sharp

contrast to the previous case, the new WIISARD system was able to deliver data from node 17. This is possible because the mobile node acts as a data mule. At position B, when nodes 9 and 17 are within communication range, node 9 will request the data from node 17. Subsequently, when node 9 returns to position A and is within the communication range of 61, it will deliver the data reports from 17 to node 61. This highlights the key advantage of the peer-to-peer architecture: data may be exchanged without requiring connectivity to the server.

Mobility Scenario:

The second experiment focuses on the impact of mobility in the absence of disconnections. In this experiment, node 61 acts the server while the remaining mesh boxes generate a new block every 30 seconds. The mesh boxes are deployed redundantly to ensure that no coverage gaps exist. A user carrying a mobile phone follows the same motion pattern as in the previous experiment.



(a) Client-server + AODV

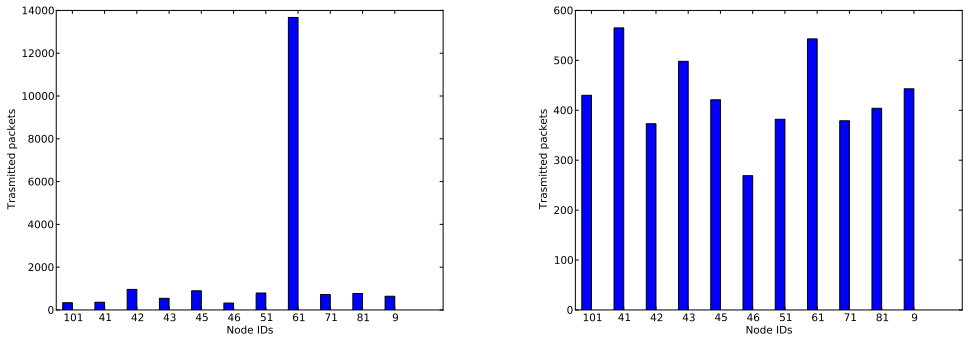
(b) Peer-to-peer + WCP

Figure 5: *Impact of mobility:* AODV performs poorly under mobility due to the need to maintain multi-hop path costs. WCP relies on local information and deliver improved reliability.

Figure 5(a) captures the impact of mobility on delivering packets from the mobile node to node 61. The figure plots two key results. First, the red circles markers indicate the time when each block was delivered. Second, the figure captures the routing failures of AODV as the red crosses.

User mobility had a profound effect on routing performance as seen by the numerous routing failures. The numerous routing failures resulted in blocks being intermittently received in short bursts. The short burst show that paths were unstable. At the end of the experiment, only 63.3% of the generated blocks of node 9 were delivered. In contrast, in Figure 5(b), we observe that the blocks from 9 were received by 61 with lower latency and higher reliability. In fact, WCP delivered 100% of the blocks by the end of the experiment which is a significant improvement. This highlights the benefit of using only local information in dynamic networking environments. It is important to note that in contrast to the previous experiment that showed a deficiency of the client-server architecture, these results highlight the limitations of ad hoc routing protocols in the presence of mobility.

Figure 6 shows the number of data packets transmitted by each node. In the case of the client-server architecture, the server transmitted as much as 15 times more packets than the other nodes in the network (see Figure 6(a)). This shows that the server can become a critical bottleneck in a client-server architecture severely limiting the scalability of such a system. The combination of the peer-to-peer architecture and WCP introduced a lower and more even distribution workload (note the different scale of the figures). When node 61 is executed from computation, WCP had on average 23% lower packets transmitted per node. This shows that the redesigned WIISARD architecture not only removes the bottleneck but also disseminates blocks more efficiently thanks to WCP's improved performance.



(a) Client-Server + AODV

(b) Peer-to-Peer + WCP

Figure 6: *Load distribution:* A client-server architecture suffers from a bottleneck at the server which must handle a large number of packets. In contrast, a peer-to-peer solution distributes the load more uniformly.

Related Work

A number of recent studies look at the impact of the mobility of emergency responses and other situational factors on wireless communication. Traditional emergency response systems adopt a client server based approach [12,13,2,14,15,16,17]. In [12] a multi-hop mesh networking architecture is proposed. Performance results and deployment problems such as difficulties in maintaining reliable connectivity and deployment optimization are discussed. The initial WIISARD system [13] provided emergency personnel and disaster command centers with medical data to track and monitor the condition of victims. As previously discussed, the system adopted a client-server architecture and multi-hop mesh networking architecture to was used to support communication among providers. Braunstein et al. [2] studyed the network behavior and feasibility of using a client server model in wireless mesh networks during disasters. They conclude that while this type of systems are beneficial, care must be taken when deploying mesh nodes in order to provide optimal coverage. As the wireless channel may fluctuate substantially during the emergency due the movement of people, vehicles and equipment, care must be taken to ensure link quality as otherwise clients may loose their connection to the server resulting in loss of collected data. A similar approach is taken in [18] but with a stronger focus on providing access and communication to off site resources through the use of Satellite, Cellular, Tetra and WLAN services. They provide performance results that show that end to end delays are strongly dependent upon the access technology used.

Having recognized the limitations of the client-server solution, peer-to-peer architectures used in conjunction with gossip protocols have been proposed as an alternative for emergency response systems [19,20,21,22]. The results presented in these paper are obtained through simulation. In contrast, in this paper we present results from a real wireless testbed which better emulates the conditions encountered during emergency responses. Moreover, while these works do provide interesting design considerations, none of these papers provide any performance comparison with traditional client bases approaches.

Bradler et al. [19] provide a communication hierarchy between different types of responders, argues for a peer-to-peer server-less network structure, and provides suggestions for how these networks may be analyzed through simulations. In [20], an Emergency Delay Tolerant Network architecture and investigates the use of SIP based communications. Bruno et. al. [21] employ an opportunistic networking approach to glue together surviving network partitions using delay tolerant approaches. Performance results that compare an optimized version of gossiping that uses context information with traditional gossiping are presented. They show that by using context information, less overhead and higher delivery can be achieved at the price of higher delay. A hybrid approach that combines traditional ad hoc routing and gossip approaches is presented in [23]. They focus on the issue of routing table stabilization, an issue during network partitioning, and use the concept of a Mediator that help coordinate and synchronize the network.

Conclusion

In this paper, we consider two approaches for ensuring the timely and reliable delivery of data during emergency responses. Empirical studies show that a client-server architecture used in combination with ad hoc routing protocols results in low reliability. These results are consistent with our experience during the deployment of the system during emergency drills. Three key factors negatively affect the reliability of the system. First, the client-server architecture performs poorly in the presence of coverage gaps that prevent clients to reaching the server. Second, the scalability of this architecture is limited by the bottleneck formed at the server. Third, the performance of ad-hoc routing protocols such as AODV is often negatively affected by user mobility. In response to these challenges, we presented a redesigned peer-to-peer architecture and novel data dissemination protocol. Empirical studies show that compared to the initial WIISARD system, the redesigned system improves reliability by as much as 36.7% while reducing the number of transmitted packets by 23%.

Acknowledgment

This work is support in part by NIH/NLM grant number R01-LM009522.

References

1. Jari Olavi Soini, Hannu Jaakkola, Jari Leppniemi, Petri Rantanen, Mika Saari, and Pekka Sillberg. Towards a distributed crisis response communication system. In *Proceedings of ISCRAM*, 2009.

2. Brian Braunstein, Troy Trimble, Rajesh Mishra, B S Manoj, Ramesh Rao, and Leslie Lenert. Feasibility of using distributed wireless mesh networks for medical emergency response. *AMIA Annu Symp Proc*, pages 86–90, 2006.

3. S. Pavlopoulos, E. Kyriacou, A. Berler, S. Dembeyiotis, and D. Koutsouris. A novel emergency telemedicine system based on wireless communication technology-ambulance. *Information Technology in Biomedicine, IEEE Transactions on*, 2(4):261 –267, 1998.

4. Tia Gao, C. Pesto, L. Selavo, Yin Chen, Jeong Gil Ko, Jong Hyun Lim, A. Terzis, A. Watt, J. Jeng, Bor rong Chen, K. Lorincz, and M. Welsh. Wireless medical sensor networks in emergency response: Implementation and pilot results. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 187 –192, May 2008.

5. Konrad Lorincz, David J Malan, Thaddeus R F Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, and Matt Welsh. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, Sep 2004.

6. M. Portmann and A.A. Pirzada. Wireless mesh networks for public safety and crisis management applications. *Internet Computing, IEEE*, 12(1):18 –25, 2008.

7. Christopher A. Kahn, Carl H. Schultz, Ken T. Miller, and Craig L. Anderson. Does start triage work? an outcomes assessment after a disaster. *Annals of Emergency Medicine*, 54(3):424 – 430.e1, 2009.

8. RFC 3561: http://www.ietf.org/rfc/rfc3561.txt.

9. Octav Chipara, Christopher Brooks, Sangeeta Bhattacharya, Chenyang Lu, Roger D Chamberlain, Gruia-Catalin Roman, and Thomas C Bailey. Reliable real-time clinical monitoring using sensor network technology. *AMIA Annu Symp Proc*, 2009:103–107, 2009.

10. Jung Woo Lee, Branislav Kusy, Tahir Azim, Basem Shihada, and Philip Levis. Whirlpool routing for mobility. In *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '10, pages 131–140, New York, NY, USA, 2010. ACM.

11. AODV-UU http://sourceforge.net/projects/aodvuu/.

12. B. Braustein, T. Trimble, R. Mishra, B.S. Manoj, L. Lenert, and R.R Rao. Challenges in using of distributed wireless mesh networks in emergency response. In *Proceedings of ISCRAM*, May 2006.

13. R.B. Dilmaghani and R.R. Rao. A wireless mesh infrastructure deployment with application for emergency scenarios. In *Proceedings of ISCRAM*, May 2008.

14. Gunnar Kramp, Kristensen, Margit Pedersen, and Jacob Frolund. Physical and digital design of the bluebio biomonitoring system prototype, to be used inemergency medical response. In *Proceedings of Pervasive Health Conference and Workshops*, November 2006.

15. Emory A. Fry and Leslie A. Lenert. Mascal: Rfid tracking of patients, staff and equipment to enhance hospital response to mass casualty events. In *Proceedings of AMIA Annual Symposium*, December 2005.

16. Tia Gao, C. Pesto, L. Selavo, Yin Chen, Jeong Gil Ko, Jong Hyun Lim, A. Terzis, A. Watt, J. Jeng, Bor rong Chen, K. Lorincz, and M. Welsh. Wireless medical sensor networks in emergency response: Implementation and pilot results. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 187 –192, May 2008.

17. S. F. Midkiff and C. W. Bostian. Rapidly-deployable broadband wireless networks for disaster and emergency response. In *Proceedings of First IEEE Workshop on Disaster Recovery Networks (DIREN '02)*, June 2002.

18. S. F. Midkiff and C. W. Bostian. Mikobos - a mobile information and communication system for emergency response. In *Proceedings of ISCRAM 2006, Third International Conference on Information Systems for Crisis Response and Management, Newark/USA*, May 2006.

19. D. Bradler and B. Schiller. Towards a distributed crisis response communication system. In *Proceedings of ISCRAM 2009, Sixth International Conference on Information Systems for Crisis Response and Management, Gotherburg, Sweden*, May 2009.

20. Fang Dong, Yuxiang Hu, Min Tong, and Xiaomin Ran. Supporting emergency service by retasking delay-tolerant network architecture. In *Proceedings of 5th International Conference Mobile Ad-hoc and Sensor Networks*, December 2009.

21. R. Bruno, M. Conti, and A. Passarella. Opportunistic networking overlays for ict services in crisis management. In *Proceedings of International Conference on Information Systems for Crisis Response and Management ISCRAM 2008*, May 2008.

22. K. Fall, G. Iannaccone, J. Kannan, F. Silveira, and N. Taft. A disruption-tolerant architecture for secure and efficient disaster response communications. In *Proceedings of International Conference on Information Systems for Crisis Response and Management ISCRAM 2010*, May 2010.

23. Thomas Plagemann, Katrine Stemland Skjelsvik, Matija Puzar, Aslak Johannessen, Ovidiu Valentin Drugan, Vera Goebel, and Ellen Munthe-Kaas. Cross-layer overlay synchronization in sparse manets. In *Proceedings of International Conference on Information Systems for Crisis Response and Management ISCRAM 2008*, May 2008.