WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2009; **9**:21–33
Published online 9 July 2008 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/wcm.653

Controlled flooding in disconnected sparse mobile networks

Khaled A. Harras[1*,†] and Kevin C. Almeroth[2]
[1]*Computer Science Department, Carnegie Mellon University, Qatar*
[2]*Department of Computer Science, University of California, Santa Barbara, CA, U.S.A.*

Summary

The incredible growth in the capabilities and functionality of mobile devices has enabled new applications and network architectures to emerge. Due to the potential for node mobility, along with significant node heterogeneity, characteristics such as very large delays, intermittent links, and high link error rates pose a new set of network challenges. Along with these challenges, end-to-end paths are assumed not to exist and message relay approaches are often adopted. While message flooding is a simple and robust solution for such cases, its cost in terms of network resource consumption is unaffordable. In this paper, we focus on the evaluation of different controlled message flooding schemes over *disconnected sparse mobile networks*. We study the effect of these schemes on message delay, network resource consumption, and neighbor discovery overhead. Our simulations show that our schemes can save substantial network resources while incurring a negligible increase in the message delivery delay.

KEY WORDS: sparse mobile networks; delay tolerant networks; epidemic routing

1. Introduction

Today's Internet operates on some overlooked assumptions such as small end-to-end round trip times (RTTs), the use of packet switching, and the existence of an end-to-end path between sources and destinations. New assumptions and challenges, however, are surfacing as different kinds of networks emerge, especially with the evolution in mobile devices. The growing dependence on these devices, along with the high mobility of users, has increased the need to be connected in all places at all times. This increase in user demand has spurred the development of numerous applications that run in new network environments. Examples of these environments include: satellite networks, planetary and interplanetary communication, military/tactical networks, sensor networks, disaster response, and other forms of large-scale mobile networks. Such environments have created a number of new challenges for network designers to solve. These new challenges include, but are not limited to, network partitioning, intermittent connectivity, large delays, high deployment cost, and the absence of an end-to-end path.

These new challenges have stimulated a great deal of research for applications such as disaster relief efforts and field hospitals [1], battlefields [2], and remote

*Correspondence to: Khaled A. Harras, Carnegie Mellon University, 5032 Forbes Av. SMC 1070, Pittsburgh, PA 15289, U.S.A.
†E-mail: kharras@cs.cmu.edu

disconnected villages [3,4], which we abstractly refer to as *disconnected sparse mobile networks* [5]. Most research, however, fails to solve many problems posed by such scenarios. Mobile ad hoc networks (MANETs) [6–11], for instance, focus more on networks where an end-to-end path is assumed to exist. On the other hand, disconnected mobile networking research [12–15] assumes some sort of control over nodes in the network, a large degree of homogeneity, or some degree of knowledge that nodes must carry regarding other nodes in the network (e.g., the path or route a node will take). Finally, delay tolerant networks (DTNs) research [5,16,17] mainly provides a generalized architecture and guidelines for addressing issues in challenged networks.

Even as network architectures evolve and as new applications emerge, message flooding remains as one of the core communication techniques. In this paper, we demonstrate how a variety of controlled message flooding schemes over sparse mobile networks affect message delay and network resource consumption. We examine the use of a probabilistic function for message forwarding. We then add a time-to-live (TTL) or kill time value on top of the probabilistic function. Finally, we contribute the novel idea of a passive cure on top of the other schemes and study its impact on the network. The passive cure is used to 'heal' the 'infected' nodes, that is, those that carry a copy of the message. In our work, we study real-life sparse mobile networks in which our work is applicable. We make the assumption that the nodes are totally blind, such that every node knows only information about itself and the messages that it carries. Another important assumption we make is that there is no form of control over any node in the network. In other words, each node is completely autonomous and makes its own decisions.

The work presented in this paper is a significant extension of our previous work [5]. This extension leads to a much greater understanding of sparse mobile networks. There are three major additions in this paper. First, we refine the system architecture and operation to be more robust, and describe the relevant algorithms in detail. Second, we present and study an alternative architecture that includes high-end stationary nodes which represent the expected widespread deployment of access points or mesh routers. Third, we conduct and present a more extensive set of simulations to evaluate our basic system as well as our alternative architecture.

The rest of this paper is organized as follows. Section 2 presents the related work. Details of our architecture, the various controlled flooding schemes, system operation, and the extended high-end architecture are all presented in Section 3. Section 4 discusses the system evaluation. Finally, the paper is concluded in Section 5.

2. Related Work

Various areas of research have tackled issues related to the new challenges of mobility, large delays, intermittent and scheduled links, high link error rates, as well as the variety of underlying network architectures and protocols. Even though MANETs addressed a subset of these challenges with a major focus on routing [6–8,10,11], it mainly assumed the existence of end-to-end paths. Due to this characteristic, these protocols are unlikely to find routes in disconnected sparse mobile networks. Hence, in this section, we briefly present some of the solutions that have been proposed to solve these challenges, namely, disconnected mobile networks and DTNs.

Most, if not all, solutions in disconnected mobile networks rely on some form of a store and forward approach. Examples of work conducted on such networks include epidemic routing [13] and dataMULEs [15]. In epidemic routing, Vahdat and Becker introduce a flooding-based routing protocol for disconnected mobile networks [13]. Nodes in the network continuously exchange copies of the messages they do not have, until the messages reach their intended destinations. Shah *et al*. [15] introduce dataMULEs, where low powered static sensors are sparsely deployed to gather various forms of data. A mobile entity, a 'mule,' then randomly travels among these sensors to gather the collected data. They also do not explore various mechanisms that may vary network resource consumption, and usually rely on simple network level flooding. Therefore, we develop various controlled flooding schemes for such needs. Spray and wait is an approach that reduces flooding overhead by spraying several message copies and routing each to the destination [18]. Our passive cure approach that we present relies on the receiver rather than predict when it will have received a message. In other words, our ultimate approach ensures message delivery, given sufficient mobility.

Considering DTNs, Fall presents a generalized DTN overlay architecture as an attempt to achieve interoperability between heterogeneous networks deployed in extreme environments [16]. These networks usually lack continuous connectivity and suffer from potentially long delays. A bundle layer protocol is introduced, where a 'custodian' assumes the

DOI: 10.1002/wcm

responsibility of reliably delivering a 'bundle' of messages to the next custodian on the path to the destination [19]. Jain *et al*. [20] expand the DTN work by studying routing issues in extreme environments. Other more specific solutions were later introduced, as in Message Ferrying [17], in order to address more specific problems such as controlling the mobility of multiple ferries used for message delivery in DTN-like networks.

The drawback of the DTN architecture [16] is that it only provides general guidelines and a framework to shed light on common problems that exist in challenged networks. Other solutions, such as message ferries [17] are more focused on the specific problem which they address. In our work, we mainly focus on studying alternatives for reducing flooding cost for cases where flooding is the only possible solution that can be adopted.

3. System Architecture

In this section, we first describe the components of our architecture, along with the functionality and behavior of each component. We then introduce the notion of '*willingness*,' which is a reflection of the degree that nodes are willing to participate in relaying messages. We then discuss the controlled flooding schemes that we propose and study in our paper. This description is followed by algorithms depicting the system operation. Finally, we present our extended architecture, an extension that incorporates high-end nodes.

3.1. Architectural Components and Assumptions

To satisfy the problems and challenges posed by the scenario offered in Section 1, we propose a transport layer overlay architecture for sparse mobile networks. Message forwarding and handling is done by this overlay layer and handles the heterogeneous protocols and node characteristics. This approach also complies with the basic DTN architecture proposed by Fall [16].

There are two important assumptions we make in our system. The first is *node blindness*. Nodes in the network do not know any information regarding the state, location or mobility patterns of other nodes. The second is *node autonomy*. Each node has independent control over itself and its movement. The reason behind these assumptions is to closely model the real world scenarios described earlier. When driving through a sparse environment, any given node has no knowledge of other nodes that come within its range (Blindness), and it is autonomous in its movement (Autonomy).

In Figure 1, we show the nodes in the network divided into three types. A *sender node*, '*S*,' is the node that initiates the transmission of a message to a destination in the network. A *forwarder node*, '*F*,' is any node that carries the message from the sender, or another forwarder, with the aim of relaying it to the ultimate node. Finally, the *ultimate node*, '*U*,' is the final destination.

The basic mechanism of node interaction is shown in Figure 1. The interaction of nodes is similar to that in epidemic routing [13], where each node continuously tries to relay a message to other nodes, within range, that do not already have the message. We look at an example where a sender node, *S*, needs to send a message.

In the beginning, *S* initiates a periodic beacon for neighbor discovery purposes. It announces that it has a message that needs to be forwarded to a specified destination, *U*. When *S* comes within range of one or more forwarder nodes, *F*, or even the ultimate node, *U*, the beacon is received and an ACK is sent to *S* from each node that received the beacon and does not have a copy of the message. When *S* receives an ACK for its beacons, it simply broadcasts the message to its neighbors. Once the message is received, the forwarder node starts to act as a sender node. It sends its own beacons, and both nodes travel through the network looking for either another forwarder to pass the message on, or for the ultimate node. The message gradually propagates through the network until it eventually reaches the ultimate node. This process results in the overuse of network resources through continuous and repetitive flooding of messages. On the other hand, the advantage of this approach is the high delivery rate and relatively small delay.

3.2. Modeling Node Willingness

Generally speaking, the frequency at which a sender or forwarder node tries to forward a message depends on many factors. Some of these factors include the node state (power or buffer space, for instance) or message state (size or priority of message). We model this as the *willingness* of a node. The willingness of a node is the degree to which a node actively engages in trying to re-transmit a message. Willingness can be modeled in terms of three variables. First, the *beacon interval* is the amount of time a sender or forwarder node waits before

DOI: 10.1002/wcm

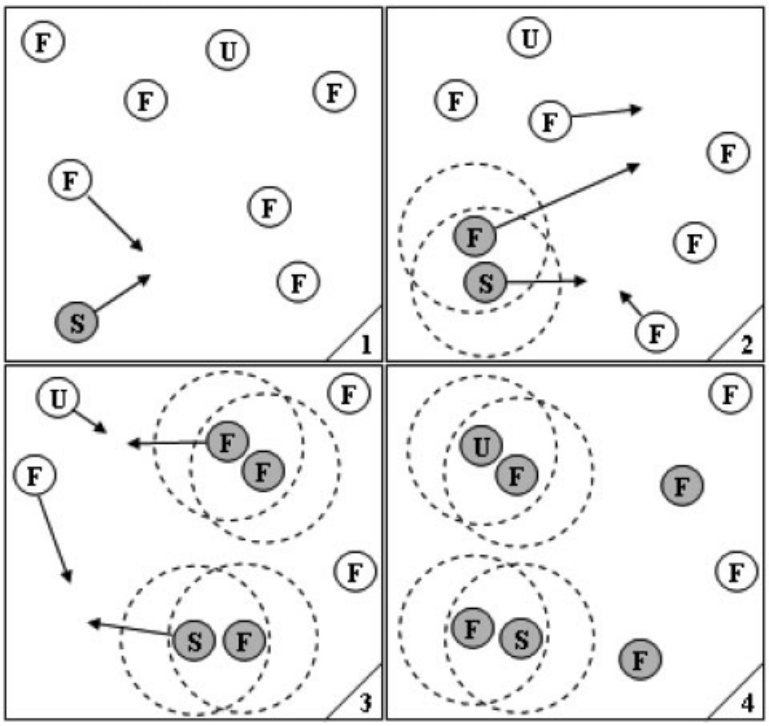

Fig. 1. Message propagation over a sparse mobile network.

sending a new beacon. Second, the *times-to-send* (TTS) is the number of times a node successfully forwards a message to other nodes in the network before it chooses to stop forwarding the message. Third, the *retransmission wait time* (RWT) is the amount of time a node waits without beaconing before it tries to resend the message to other nodes in the network. The source node includes the value of these parameters as part of the message header. This way, the forwarder nodes can set their willingness levels accordingly.

To help clarify how these three variables affect the behavior of a given node in the network, we introduce the following simple example. Let us assume that the beacon interval = 1 s, TTS = 2 s, and the RWT = 50 s. These values indicate that when a sender wants to send a message, it sends a beacon every second to find other nodes that would carry the message. Once the sender node finds a forwarder node, by receiving an ACK for its beacon, it transmits the message and decrements the times-to-send by one. The sender then waits for 50 s before it resumes sending beacons every second to look for the next node to which it will forward the message. This process repeats until the TTS reaches zero. The forwarder nodes that received the message in both cases start acting as the original sender, assuming that all nodes have the same willingness.

3.3. Specific Controlled Flooding Schemes

In this paper, we study different controlled flooding schemes. We are careful to base our schemes on simple, non-chatty, and elegant algorithms. We choose this goal, because, in sparse and highly mobile networks, complex or chatty algorithms waste the short time nodes have when they come within range of each other. The schemes we introduce are the following:

(1) **Basic probabilistic (BP):** When describing the *willingness* of a node in the previous section, we implied that forwarder nodes have the same willingness as the sender node. To more closely emulate reality, however, we choose a uniform probability distribution that determines the willingness of the nodes to transmit a given message. Based on the result of this function, a forwarder may choose not to forward the message at all, forward it at half the willingness of the sender, or forward it at the same level of willingness as the sender.

(2) **Time-to-live (TTL):** In this scheme, we add a TTL value. The TTL here determines how many times the message is forwarded before it is discarded. We add the TTL on top of the BP scheme since

DOI: 10.1002/wcm

the BP scheme is a more realistic representation of how nodes act regarding the choice to forward messages.

(3) **Kill time:** Here, we add a timestamp to the message on top of the BP scheme. The timestamp is the point in time after which the message should no longer be forwarded. The timestamp is an absolute universal life time for the message. This technique would be appropriate if, for example, the sender node knows how long it will be disconnected. This technique is also a good way to set the maximum time a node should keep a message in its buffer if the TTS variable of that message does not reach zero.

(4) **Passive cure:** The final scheme, or optimization, we introduce is the novel idea of the *passive cure*. The idea is that, once the ultimate node receives the message, it generates a passive cure to 'heal' the nodes in the network after they have been 'infected' by the message. The ultimate node 'cures' the forwarder that passed the message to it by sending a 'cure-ACK' instead of an ordinary ACK that is sent when a beacon is received. Whenever a forwarder or the ultimate node detects any other node sending the same message, it sends a cure-ACK to that node to prevent future retransmissions.

3.4. System Operation

In this section, we offer a more detailed description of our system's operation of our system as well as the specific control flooding schemes we are studying. We do so by presenting the algorithms that represent the operation of sending a single message from a source node to the ultimate node in a sparse mobile network. These algorithms focus on the operation of the forwarder nodes, since their operation is the determining factor for how the system operates as a whole. We assume that the passive cure is applied to the system along with the other stopping techniques. The algorithms are shown in Figures 2–5.

Figure 2 describes the general notation that will be used for all the algorithms in this section (Lines 1–4), as well as the forwarder node setup and initialization (Lines 5–8). All forwarder nodes begin as 'uninfected' with respect to the message that needs to be delivered (Line 7). The willingness level of each forwarder is then set according to the probability function we introduced earlier (Line 8). Meanwhile, the source is now trying to send a message to the ultimate destination.

Figure 3 depicts the operation of an uninfected forwarder. As long as the node is uninfected, it roams around until it receives a beacon from the source or another infected forwarder (Lines 9–11). If the willingness of this forwarder is not zero, it acknowledges the beacon and receives the message (Lines 11–13). If the TTL value on the received message reaches zero, the message is then dropped and the forwarder continues to move without being infected (Line 14). Assuming the TTL is not zero, the forwarder is then infected with the message, sets the number of times it needs to forward the message (the TTS value) according to its willingness level, and updates the message kill time (Lines 15–18).

```
1: // Notation
2: S: Sender, U: Destination, F: Forwarder, B: Beacon
3: m: Message, n: Total # of Nodes, W: Willingness
4: →: Sends, ←: Receives, tts: Times-To-Send

5: // Forwarder nodes setup and initialization
6: for every F_i where i goes from 1 to n-2 do
7:     F_i.state_m = Uninfected;
8:     W_Fi = f_probabilistic(n);
```

Fig. 2. Notation and system startup.

```
9: while (F_i.state_m == Uninfected) do
10:    F_i moves according to mobility model;
11:    if ((F_i ← B_m from F_j or S) && (W_Fi ≠ 0)) then
12:        F_i → ACK_Bm to F_j or S;
13:        F_i ← m from F_j or S;
14:        if (− − m_TTL == 0) then drop m and continue;
15:        F_i.state_m = Infected;
16:        F_i.tts_m = W_Fi(m);
17:        if (Kill_Time == True) then
18:            Update Kill_Time_m accordingly;
```

Fig. 3. Uninfected forwarder node operation.

```
18: while ((F_i.state_m == Infected) &&
19:            (F_i.tts_m > 0) && (Kill_Time_m > 0)) do
20:    F_i moves and broadcasts B_m;
21:    if (F_i ← ACK_Bm from F_j) then
22:        F_i → m to F_j;    F_i.tts_m = F_i.tts_m − 1;
23:    if (F_i ← ACK_Bm from U) then
24:        F_i → m to U;
25:        F_i ← ACK_Cure from U;
26:        F_i.state_m = Cured;    F_i.tts_m = 0;
27:    if (F_i ← B_m from F_j) then ignore;
28:    update Kill_Time_m;
29: // We examine why the while loop ended
30: if (((F_i.tts_m == 0) || (Kill_Time_m == 0))
31:    && (F_i.state_m ≠ Cured))
32:        then F_i.state_m = Uninfected;
```

Fig. 4. Infected forwarder node operation.

DOI: 10.1002/wcm

```
33: while ((F_i.state_m == Cured) && (Cure_T ≠ 0)) do
34:     F_i moves according to mobility model;
35:     if (F_i ← B_m from F_j or S) then
36:         F_i → ACK_Cure to F_j or U;
37:         update Cure_T;
38: if (Cure_T == 0) then F_i.state_m = Uninfected;
```

Fig. 5. Cured forwarder node operation.

Figure 4 illustrates the operation of an infected forwarder. In general, the infected forwarder keeps moving and tries to pass the message to other uninfected forwarder nodes, or to the ultimate node as long as the kill time and TTS values for the message do not reach zero (Lines 18–20). Once the infected forwarder receives an acknowledgment for its beacon from an uninfected forwarder, it forwards the message and updates the TTS value (Lines 21–22). If the uninfected node happens to be the ultimate node, the infected node then receives a cure acknowledgment and sets its state to cured (Lines 23–26). Meanwhile, the infected forwarder ignores any beacons it might receive for the message it is currently trying to send, and occasionally updates the message kill time (Lines 27–28). Finally, if the while loop ended without the forwarder being cured, the message is dropped and the forwarder returns to the uninfected state (Lines 30–32).

The last state in which a forwarder node can operate, the cured state, is shown in Figure 5. The forwarder in this case moves around waiting for other infected nodes to initiate contact with it, so long as the cure time does not expire (Lines 33–34). The cure time is a timer chosen sufficiently large, after which the cured state can be deleted by the node. Once the cured forwarder receives a beacon from an infected forwarder or the source, it sends its cure acknowledgment in order to heal the infected node, and updates its cure time (Lines 35–37). Once the cure time expires, the forwarder returns to its initial uninfected state (Line 38).

3.5. Extended Architecture With H-nodes

The expected deployment of access points and mesh routers, along with a study showing that mobile devices can experience periods of connectivity with high throughput, even at speeds of 75 mph [21], leads us to consider a modified architecture. We extend our architecture to include what we call 'H-nodes.' H-nodes are basically high-end nodes fully connected to each other through wired or wireless backbones. These H-nodes represent deployed access points or mesh routers that may exist in some portions of a sparse mobile network. H-nodes are therefore stationary nodes characterized by having higher transmit power and storage capacities.

After studying various controlled flooding schemes, we choose the most promising combination of schemes that give the best performance based on the results shown in Section 4. We use this scheme in our extended architecture and study the impact of having these high-end nodes in our system. Figure 6 then demonstrates how the extended architecture operates. The figure is based on using a combination of passive cure + TTL + BP to control message flood in the network. This combination has proven to work best according to our simulation results.

Figure 6 shows three H-nodes deployed and fully connected to each other. Other mobile nodes move in the network. When S has a message to send, it passes it to the first forwarder node that comes within range. When F comes within range of any H-node, all the interconnected H-nodes are infected by the message. These H-nodes consequently infect other forwarder nodes that are within their range. Once U receives the message, it sends its 'cure' to heal the forwarder that gave it the message. The cured forwarder then moves toward one of the H-nodes thereby curing it along with all the other H- and F-nodes within their range. In this new architecture, the ultimate destination may be one of the H-nodes. This condition occurs, because, in many cases, the email or transaction that needs to be sent, simply needs an access point or mesh router connected to the Internet. If U is a node in the network, then the H-nodes will help to deliver the message faster, as shown in Figure 6.

In general, if U is not one of the H-nodes, then the H-nodes could act in several different ways based on their unique characteristics. First, they could actively forward the message onto other forwarders, thus, acting the same way a forwarder node would act, but with full willingness. Second, the H-nodes could initiate the passive cure, and be responsible for passing the message only when the U node comes within its range. The advantage of this technique is that fewer nodes will be infected, but the delay might be higher. Finally, they could use high power signals, in either of the previous two modes, to send the message over larger ranges.

The goal of introducing this extended architecture is to demonstrate how a simple disconnected sparse mobile network could evolve to be a more realistic combination of ad hoc and/or mesh network, with potentially numerous Internet gateways. In any of

DOI: 10.1002/wcm

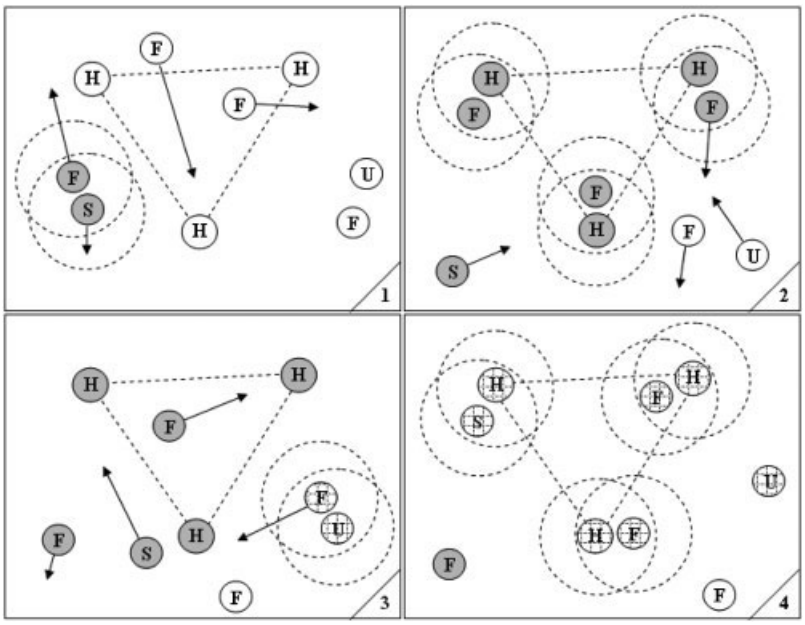

Fig. 6. Extended architecture with H-nodes.

these architectures, our algorithms are designed to be efficient, effective, and elegant.

4. Evaluation

In our evaluation, we seek to achieve three main goals. First, we hope to analyze how a controlled flooding scheme behaves in general. Second, we want to measure how the specific flooding schemes affect network efficiency and overall delay. Finally, we examine the impact of the enhanced H-node architecture on our metrics. We now describe our simulation setup and environment and follow it by summarizing the outcomes of an extended set of simulations.

4.1. Simulation Environment

We conducted our simulations using the GloMoSim network simulator. We added an overlay layer that handles all the message bundling and relaying, and implements the controlled flooding schemes that we have described. Since we do not have real movement data for our target scenarios, we use a *modified* random way-point mobility model that avoids the major problem of node slow down in the conventional random way-point model [22]. Even though there are other more realistic mobility models that have been developed, we believe that they do not accurately model the scenarios we are concerned with. Therefore, we adopt the simple random way-point model since we believe as well that the mobility model will have little impact on the 'relative' performance of our schemes. The node speed ranges between 20 and 35 m/s; and the rest period is between 0 and 10 s. Finally, every point in our results is taken as an average of 10 different seeds.

Table I. Simulation parameters.

Parameter	Value range	Nominal value
Number of nodes	25–200	100
Terrain	10–50 km^2	10 km^2
Simulated time	1–24 h	6 h
Transmission range	250 m	250 m
Beacon interval	0.5–50 s	1 s
Times-to-send (TTS)	1–50	10
Ret. wait time (RWT)	0–500 s	50 s
Time-to-live (TTL)	2–10	7
Kill time	1000–21 600 s	5000 s
Number of H-nodes	1–10	1

The major parameters used in our simulations are summarized in Table I. The *terrain* is the area over which the *number of nodes* are scattered. Each node has a *transmission range* of 250 m. *Simulated time* represents the amount of time the simulations run. The *beacon interval* is the period after which beacons are sent. A 'beacon' is simply a signal emitted by all nodes to search for other nodes in the network as well as to announce its location. The TTS is the number

DOI: 10.1002/wcm

of times a node will successfully forward a message to other nodes in the network. *Retransmission wait time* represents the amount of time a node remains idle after successfully forwarding a message to another node. When the retransmission wait time expires, the node then tries to resend the same message. We mainly use TTS to represent the willingness of the nodes to participate in message relaying. The TTL is the number of hops after which a message is discarded. *Kill time* is the amount of time after which the message is discarded. Finally, the *number of H-nodes* are the number of high-end nodes scattered in the system.

We consider two metrics in evaluating our schemes. First, *network efficiency* is represented by the total number of messages sent by the nodes in the network. Second, *overall delay* is the total time that elapses from when a node wants to send a message until the ultimate node receives that message for the first time. We note that the results are shown for a 100 per cent delivery ratio, since we want to see the cost of delivering a given number of messages in terms of our metrics.

4.2. Results

Before applying our probabilistic scheme, we analyze how the network acts assuming full willingness of all the nodes in the network along with enforcing some basic level of message control. This scheme is very similar to the epidemic routing scheme, but with some control over message forwarding. We use the TTS variable to represent node willingness. We investigate how varying the TTS value and the network density affects our metrics. The RWT in these experiments is 1 s.

Figure 7(a) shows the total number of messages sent by all the nodes in the network and Figure 7(b) shows the delay. Overall, Figure 7 shows that increasing the network density results in an increase in the number of messages and decrease in delay, since more nodes are sending more copies of the message. One interesting point is that an increase in the network density results in a significant decrease in the overall delay only up to a certain point (number of nodes = 100), after which, the decrease in delay is overshadowed by the corresponding increase in cost. On the other hand, increasing the willingness beyond a certain point (TTS = 10) does not have a large effect on either metric. The reason for this result is that the nodes in such sparse networks do not encounter each other often enough to consume the large value of the TTS.

4.2.1. Basic probabilistic behavior

After applying our BP scheme, we examine how the network density and RWT impacts our metrics. We assume that 25 per cent of the nodes in the network have zero willingness, 25 per cent have full willingness, and 50 per cent of the nodes forward the message with only half the willingness (i.e., half the TTS of the source node).

Figure 8 shows the result of varying the network density while keeping the TTS set to 10. One interesting observation is the drop in terms of the total number of messages when the RWT increases, with a corresponding small increase in delay (until RWT reaches 100). With a small RWT, the message spreads rapidly through the network, and in a very short time, many forwarders are actively trying to send the message. As the RWT increases, the message does not initially reach as many forwarders. We witness a drop in results due to this behavior. Also, with a large RWT, the TTS is not consumed as quickly, so a forwarder node acts as a forwarder longer, thus rejecting receipt of the

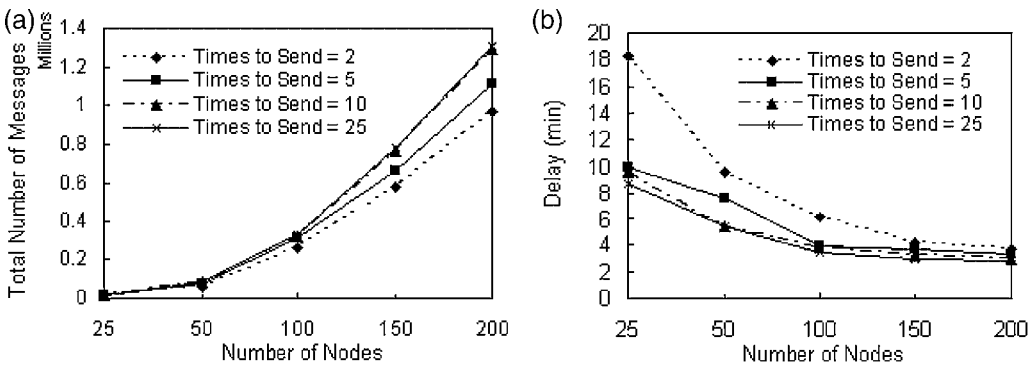

Fig. 7. The impact of changing the number of nodes and times-to-send (TTS) on (a) total number of messages and (b) overall delay in the basic scheme.

DOI: 10.1002/wcm

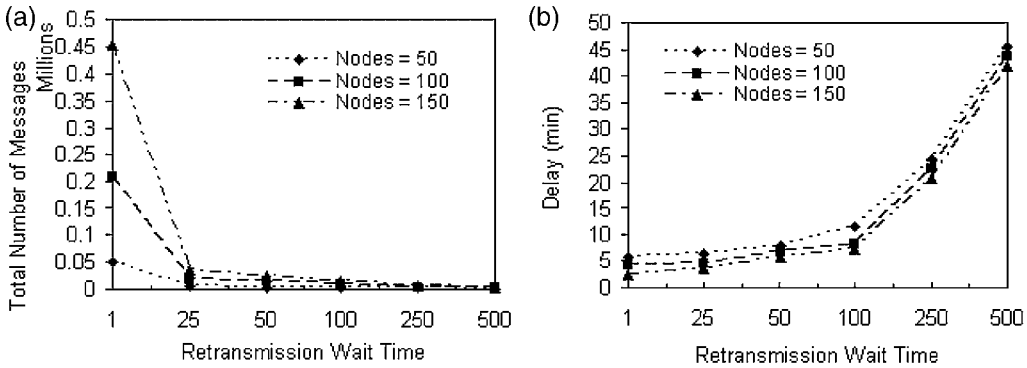

Fig. 8. The impact of changing the retransmission wait time (RWT) and number of nodes on (a) the total number of messages and (b) overall delay.

same message. With a small RWT, the TTS is quickly consumed. Because the nodes do not keep any state or cache, they are ready to receive the same message and start forwarding it again.

Similar result patterns are generated when keeping the number of nodes constant, but while varying the TTS. We have also seen similar patterns when we used only the basic controlled scheme, but with a much larger scale in terms of the total number of messages. This result shows how a uniform probability based scheme performs much better. The results are not shown due to space limitation.

4.2.2. *Time-to-live (TTL) and kill time*

Keeping the basic probabilistic scheme, we build the other schemes on top of it to see their impact on our metrics. Figure 9(a) shows the impact of adding TTL only and adding TTL + passive cure, to the basic probabilistic scheme. We discuss the passive cure results later, and here focus on the TTL.

The first result we observe in Figure 9(a) is the large decrease in the total number of messages sent in the network (note the scale is significantly smaller than in Figures 7(a) and 8(a)). This result occurs because the TTL puts a limit on the number of times a message is forwarded. Previously, messages endlessly propagated throughout the network with no limit other than the nodes' willingness.

The impact of TTL on delay is not shown here. However, we found that as the TTL increases, the overall delay decreases up to a certain point (at TTL = 7), after which it remains relatively constant at 10 min. Even though there is a probability of message loss when using low TTLs, we believe it is a better choice when added to the BP scheme. If a large TTL is set, our simulations show that all messages reach the destination, and the cost is much smaller than in the BP scheme. For instance, if we choose a TTL of 9, we incur a cost of 1000 messages instead of 25 000 messages, while keeping the overall delay the same for both cases.

Figure 9(b) shows the impact of adding only the kill time mechanism, and adding kill time + passive cure, to the basic probabilistic (BP) scheme. The kill time scheme introduces another improvement over the BP scheme alone. The use of a universal time after

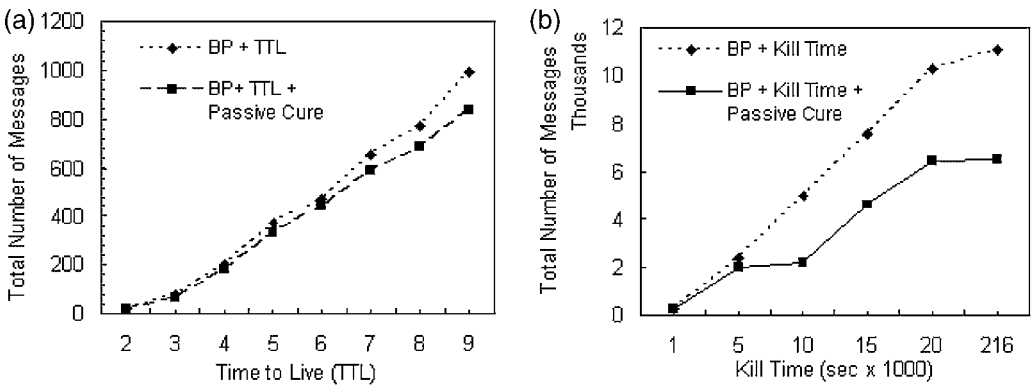

Fig. 9. The impact of adding (a) TTL and passive cure to BP and (b) kill time and passive cure to BP.

DOI: 10.1002/wcm

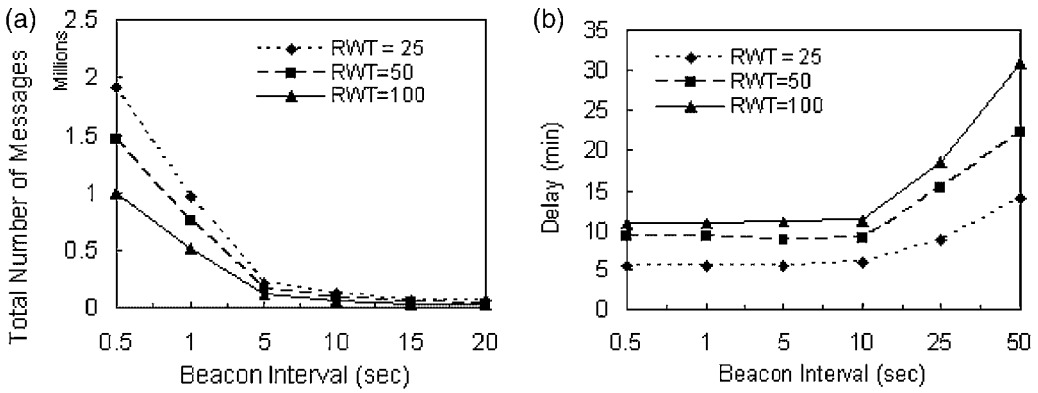

Fig. 10. The impact of the beacon interval on (a) the total number of beacons and (b) the overall delay.

which a message is discarded certainly stops message transmission and propagation. Figure 9(b) shows how the total number of messages increases as the kill time is increased. On the other hand, the overall delay (not shown due to space limitation) remains constant at 10 min. The only disadvantage of the kill time is if it is set too small so that the message does not make it to the destination.

4.2.3. Passive cure

In Figures 9(a) and 9(b), we also demonstrate the effect of adding the passive cure optimization to the TTL and kill time, respectively. The passive cure does not introduce any improvement in terms of delay because it does not help the message get to the ultimate node any faster. The effect of the cure is evident only in the total number of messages sent in the network. As Figure 9 shows, when the passive cure is added, there is a drop in the total number of messages. This drop is due to the fact that when the cure starts to operate, the cured nodes stop trying to forward the messages even if the kill time has not been reached or if the TTL has not been consumed.

The passive cure optimization has several advantages over the other schemes. First, if the message reaches the ultimate node early, little network flooding may take place since the cure stops the flood early on. Second, the ultimate node receives the message only once. In all the other schemes, however, the ultimate node may receive the same message multiple times. Finally, the passive cure may be used as a way to implement end-to-end acknowledgments if it is forced to propagate back to the sender node, since the sender would then know that its message reached the ultimate node.

4.2.4. Beacon interval

In order to perform all the message exchanges in a sparse mobile network, neighbor discovery is of great importance. A node occasionally broadcasts beacons in order to announce its location and search for other nodes that may be within its range. The frequency of these beacons has a great impact on the metrics we consider in our system. This impact is illustrated in Figure 10, where we show results for the impact of the beacon interval time on the number of beacons sent and the total delay.

The overall results in Figure 10 show, in general, that smaller beacon intervals result in better neighbor discovery which leads to a larger number of messages (not shown) and beacons being sent. This result leads to a faster spread of the message in the network which results in a lower overall delay. The interesting observation, however, is the fact that a small change in the beacon interval can lead to a large decrease in cost with minimal impact on delay. This fact is clearly shown for the beacon interval between 0 and 10 s. We can see a large drop in terms of the number of beacons (Figure 10(a)), with almost no increase in the average delay (Figure 10(b)). The reason for this behavior is that, for any given mobile system, depending on the mobility and density of the network, there will be a threshold in terms of beacon interval time, below which no further gain can be achieved.

4.2.5. Scheme comparison

In this section, we compare the performance of all the flooding schemes. Overall, Figure 11 shows the performance of these schemes relative to each other. When looking at Figures 11(a) and 11(c), we observe that the BP scheme by itself is the most expensive,

DOI: 10.1002/wcm

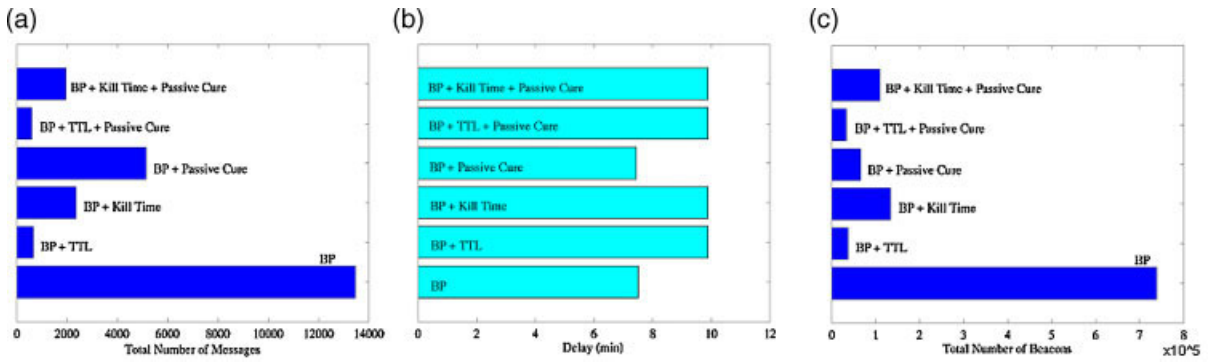

Fig. 11. Comparing the impact of the controlled flooding schemes on (a) the total number of messages, (b) the overall delay, and (c) the total number of beacons.

while the BP + TTL + passive cure is the least expensive in terms of the total number of messages and the total number of beacons sent. Note that the BP scheme already performs better than the basic flooding technique, which is analogous to epidemic routing.

Figure 11(b) shows that most of the schemes have a similar overall delay. The advantage is for the BP and BP + passive cure schemes. The reason for this result is that the complexity introduced by the TTL or the kill time results in an overall increase in delay. However, a 2 min increase, with a corresponding large decrease in the number of messages and beacons sent, certainly seems acceptable.

Figure 11 summarizes the general tradeoffs between the schemes we introduce. For example, adding the passive cure to the BP scheme saves about 60 per cent of the total number of messages, with no increase in the overall delay. When adding the TTL to the BP scheme, the total number of messages is reduced by more than 90 per cent, while the overall delay increased by only 2 min. This increase in delay does not notably affect most of the applications we envision.

4.2.6. *Impact of H-nodes*

After analyzing Figure 11, we choose the passive cure + TTL + BP as our preferred scheme in sparse mobile networks. We use this scheme for our tests on the extended H-node architecture. We show results for sparse networks, setting the number of nodes to 25 and 50. We simulate cases where H-nodes are assumed to represent ultimate nodes. In other words, the goal of the S node is to get the message to any H-node in the sparse network.

Figure 12 shows the impact of varying the number of H-nodes in a network of 25 and 50 nodes on our metrics. Both Figures 12(a) and 12(b) depict a general trend of a decrease in the total number of messages and overall delay as the number of H-nodes is increased. This result is due to the fact that, with a larger number of H-nodes, the message has a better chance of reaching an AP, and so, the message can be delivered faster. The total number of messages decreases, because when the number of H-nodes increases, we have more nodes that can 'simultaneously' heal the network using the passive cure.

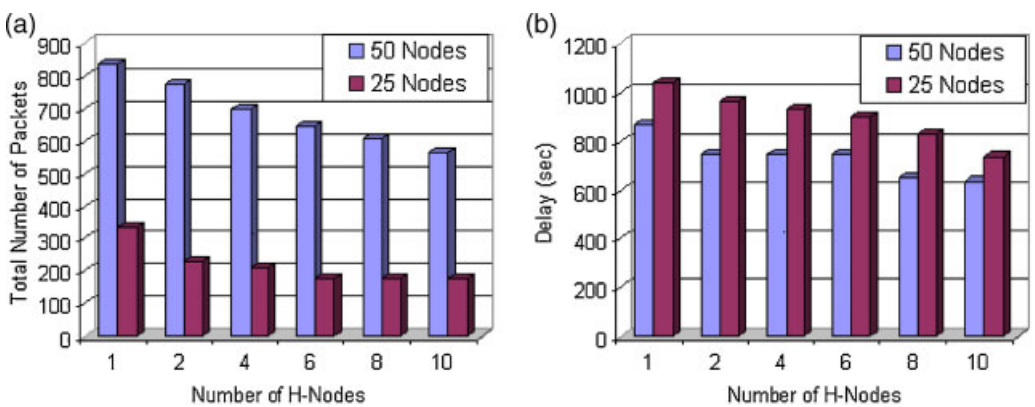

Fig. 12. The impact of changing the number of H-nodes on (a) the total number of messages and (b) overall delay.

DOI: 10.1002/wcm

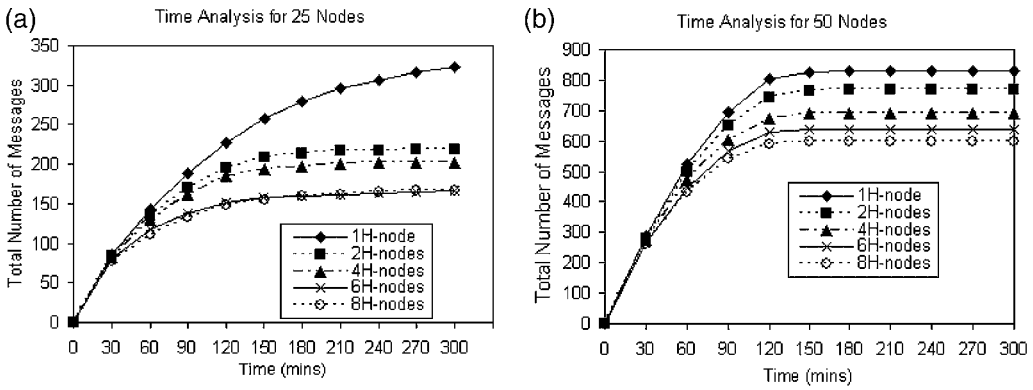

Fig. 13. Time series analysis of the total number of messages when using H-nodes with (a) 25 nodes and (b) 50 nodes.

More interesting results can be observed in Figure 13. The figure generally shows a time series analysis of the total number of messages that have been sent in the network at specific points in time. Figures 13(a) and 13(b) show the impact of changing the number of H-nodes on the cumulative total number of messages sent at a given time in networks with 25 and 50 nodes, respectively.

Generally speaking, the message spreads faster in denser networks. This result can be seen by the sharper increase in the total number of messages in the 50 node network. Also, the network heals faster in denser networks. This result can be seen by the faster convergence of lines in the 50 node network when compared to that of the 25 node network. This convergence is due to the healing of the network, after reaching a point where all the nodes are healed (or still infected but cannot find anyone to pass the message to), and no more messages are sent through the network. Finally, we observe the intuitive behavior of how increasing the number of H-nodes would generally decrease the total number of messages that are injected into the network. This result occurs due to the increase in the number of high-end nodes that heal the network at a higher rate.

5. Conclusion

In this paper, we have studied the problem of efficient message delivery in delay tolerant sparse mobile networks. We have proposed several controlled flooding schemes on top of a transport layer overlay architecture. The specific schemes we have examined are BP, TTL, kill time and passive cure. We have studied the impact of these schemes on network efficiency and overall message delivery delay. Our simulations have demonstrated that for a given sparse mobile network, the schemes reduce the number of messages and beacons sent in the network. This occurs with either no increase or only a small increase in the overall message delay. We then chose the scheme that performs best, which turned out to be a combination of the passive cure with TTL and BP schemes, and stress tested it. We have examined this combination scheme over an extended architecture that we have proposed, to accommodate for other real-world scenarios.

With the completion of our work, we believe that we have significantly added to the literature on disconnected mobile networks, and have opened several new directions of research. Our future work includes addressing security issues in such scenarios since such networks are subject to denial of service attacks. Also, since flooding-based schemes do not perform well in dense environments, we intend to develop measures to help nodes modify their behavior when they enter densely populated areas.

References

1. University of South Florida: Center for Robot-Assisted Search and Rescue. http://crasar.csee.usf.edu/
2. Krotkov E, Blitch J. The defense advanced research projects agency (DARPA) tactical mobile robotics program. *The International Journal of Robotics Research* 1999; **18**(7): 769–776.
3. Technology and Infrastructure for Emerging Regions. http://tier.cs.berkeley.edu/
4. First Mile Solutions. http://www.firstmilesolutions.com/
5. Harras K, Almeroth K, Belding-Royer E. Delay tolerant mobile networks (DTMNs): controlled flooding schemes in sparse mobile networks. In *IFIP Networking*, Waterloo, Canada, May 2005.

DOI: 10.1002/wcm

6. Johnson D, Maltz D. *Dynamic Source Routing in Ad Hoc Wireless Networks*, Vol. 353. Kluwer Academic Publishers: Dordrecht, 1996.
7. Perkins C. Ad-hoc on-demand distance vector routing. In *IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999; pp. 90–100.
8. Haas Z, Pearlman M. The performance of query control schemes for zone routing protocol. In *ACM SIGCOMM*, Vancouver, Canada, August 1998; pp. 167–177.
9. Royer E, Toh C. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine* 1999; **6**(2): 46–55.
10. Perkins C, Bhagwat P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM*, London, England, October 1994; pp. 234–244.
11. Ko Y-B, Vaidya N. Location-aided routing (LAR) in mobile ad hoc networks. In *ACM MobiCom*, Dallas, TX, October 1998; pp. 66–75.
12. Li Q, Rus D. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *ACM MobiCom*, Boston, MA, August 2000; pp. 44–55.
13. Vahdat A, Becker D. Epidemic routing for *partially connected* ad hoc networks. *Technical Report CS-200006*, Duke University, April 2000.
14. Juang P, *et al*. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, CA, October 2002.
15. Shah R, Roy S, Jain S, Brunette W. Data MULEs: modeling a three-tier architecture for sparse sensor networks. In *IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, AK, 2003.
16. Fall K. A delay-tolerant network architecture for challenged internets. In *ACM SIGCOMM*, Karlsruhe, Germany, August 2003.
17. Zhao W, Ammar M, Zegura E. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *IEEE INFOCOM*, Miami, FL, March 2005.
18. Spyropoulos T, Psounis K, Raghavendra C. Efficient routing in intermittently connected mobile networks: the multi-copy case. *IEEE/ACM Transactions on Networking* 2008; **16**(1): 77–90.
19. Fall K, Hong W, Madden S. Custody transfer for reliable delivery in delay tolerant networks. *Intel Research, Berkeley-TR-03-030*, July 2003.
20. Jain S, Fall K, Patra R. Routing in a delay tolerant network. In *ACM SIGCOMM*, Portland, OR, August 2004.
21. Gass R, Scott J, Diot C. Measurements of in-motion 802.11 networking. In *Workshop on Mobile Computing Systems and Applications*, Semiahmoo Resort, WA, April 2006.
22. Bettstetter C, Hartenstein H, Perez-Costa X. Stochastic properties of the random waypoint mobility model. *Wireless Networks* 2004; **10**(5): 555–567.

Authors' Biographies

Khaled A. Harras is currently an Assistant Teaching Professor in the Department of Computer Science at Carnegie Mellon University. He currently resides at CMU's campus in Qatar. Khaled's research focuses on delay and disruption tolerant networks, specifically protocol and architectural challenges in extreme networking environments. He is also interested in wireless and mobile ad hoc networks, particularly vehicular and sensor networks. He is a member of IEEE and the ACM.

Kevin C. Almeroth is currently a Professor in the Department of Computer Science at the University of California in Santa Barbara where his main research interests include computer networks and protocols, wireless networking, multicast communication, large-scale multimedia systems, and mobile applications. He has published extensively with more than 150 journal and conference papers. He is also heavily engaged in stewardship activities for a variety of research outlets including journal editorial boards, conference steering committees, new workshops, and the IETF. He is a Member of the ACM and a Senior Member of the IEEE.

DOI: 10.1002/wcm