Six-shot Broadcast: a context-aware algorithm for efficient message diffusion in MANETs

Benoît Garbinato, Adrian Holzer, and François Vessaz

Université de Lausanne, Lausanne, Switzerland
{Benoit.Garbinato,Adrian.Holzer,Francois.Vessaz}@unil.ch
http://www.hec.unil.ch/dop

Abstract. In this paper, we introduce *six-shot broadcast (6SB)*, a new context-aware message diffusion algorithm that uses location information to fine-tune its broadcasting process. Message diffusion is indeed one of the core challenges brought up by distributed systems and has therefore largely been studied in the context of traditional network structures such as the Internet. With the emergence of mobile ad hoc networks *(MANETs)*, new broadcasting algorithm especially geared at these networks have been introduced. These algorithms must reach two conflicting objectives when broadcasting a message, namely reliability vs. efficiency. That is, they must maximize the number of nodes that deliver the message (reliability), while minimizing the number of nodes that forward the message (efficiency). In recent years as more and more mobile devices have become context-aware, several broadcasting algorithms have been introduced using contextual information, such as location, in order to increase reliability and efficiency. Along that line, we provide a in-depth performance evaluation of our 6SB algorithm, by comparing it to similar broadcasting algorithms also targeted at *MANETs*. Our results show that *6SB* competes with the most efficient algorithms in high densities of nodes and offers increased reliability in low densities at a reasonable overhead.

Key words: MANETs, broadcast algorithm, context-aware systems.

1 Introduction

During the past years we have been witnessing a massive increase of mobile devices. These devices are now ubiquitous and changed the traditional distributed systems from centralized and wired architectures to dynamic, heterogeneous and frequently changing network architectures like mobile ad hoc networks *(MANETs)*. These architectures offer many new opportunities for application developers and new challenges for networking protocol designers. The former develop new types of dynamic mobile applications used for example in traffic jam prevention, information dissemination in crowds, strategic data gathering in hostile environments, or peer-to-peer mobile games. The later focus on designing low-level protocols such as broadcast, consensus or atomic commit to create the building blocks for the application developers. In this paper, we introduce *six-shot broadcast (6SB)* as one of these building blocks.

1.1 Mobile ad hoc networks

Mobile ad hoc networks *(MANETs)* are networks of mobile devices without fixed infrastructure. In such a network, every node is directly connected to all nodes located within the range of its radio signal, also referred to as the *technical range*. These nodes are called *neighbors*. Since nodes are mobile, neighborhoods change over time as nodes get in and out of each others technical range.

Communication between neighbors is trivial, as every message emitted by a node is received by all neighbors, thus no routing is necessary. Communication with nodes located outside the neighborhood is a more interesting aspect of *MANETs*, since it implies *multi-hop* message diffusion, where the message must be forwarded by one or more intermediate relays between the sender and the receiver. Henceforth, we will only consider multi-hop message diffusion.

1.2 Message diffusion in MANETs

In this paper, we address one kind of message diffusion, namely broadcast, where one node sends a message to all nodes in the network. Broadcasting in *MANETs* is made difficult by three of its inherent properties: (1) frequent network configuration change due to the mobility of devices, (2) decentralized architecture due to the absence of fixed infrastructure, and (3) limited resources due to small mobile devices. Such resources include battery and CPU power and the limited network bandwidth, which favors message collisions and might lead to what is called a *broadcast storm* [7] paralyzing the network. In order to evaluate and compare different broadcasting algorithms, two dimensions can be considered:

Reliability indicates the number of nodes in the network that deliver a message compared to the number of nodes that should have delivered the message. We measure this dimension via the *delivery ratio*. The higher the delivery ratio, the higher the reliability.

Efficiency indicates the cost of broadcasting in terms of message retransmissions. This dimension is measured by the *forward ratio*, the number of nodes relaying a message divided by the total number of nodes in the network. The higher the forwarding ratio, the poorer the efficiency.

The challenge of broadcasting in *MANETs* is to ensure that a high number of nodes deliver the message (reliability), while a small number of nodes retransmit this message (efficiency).

1.3 Contribution and roadmap

In Section 2, we present a novel context-aware broadcasting algorithm called *six-shot broadcast (6SB)* which constitutes the main contribution of this paper. Section 3 then discusses related work before Section 4 presents the second contribution of the paper, a thorough performance evaluation and comparison of *6SB* and related algorithms. Section 5 wraps up this paper with concluding remarks and hints on future research opportunities.

2 Six-shot Broadcast

The *six-shot broadcast (6SB)* algorithm is a *context-aware* broadcasting algorithm. which uses location as context information. The *6SB* interface offers the two following typical primitives:

6SB-BROADCAST(m): broadcasts the message m to all nodes located in the network.
6SB-DELIVER(m): acts as a callback when message m is received.

The particularity of *6SB* resides in the scheme it uses to decide whether or not to forward a given message. This scheme is based on a widespread mechanism that we dub *wait and see*. With this mechanism, when a message m is received, a node initiates a waiting time during which it looks for retransmissions of m. When the waiting time elapses m is forwarded unless the number of retransmissions seen is higher than a predetermined *threshold*. *6SB* assigns a different waiting time depending on the geographical location of nodes. The idea is that before a node sends a message m, it associates six geographical *targets* to it as depicted in Figure 1. Then among all nodes that receive m, only those located closest to a target should forward m. Note that only nodes located in what is called the *forward zone* can possibly forward messages, all nodes located in the *no forward zone* never forward a message.

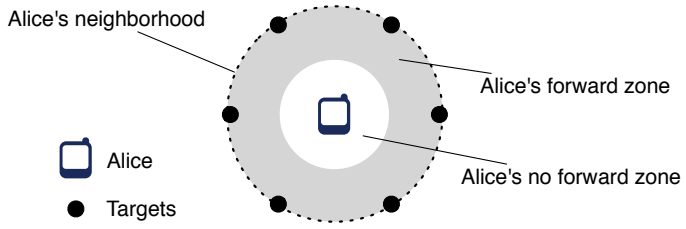

Fig. 1. 6SB principle

2.1 Underlying services

6SB is built as a primitive that can be used by the application layer and uses two underlying services. As it is location-based it uses an underlying *Location Service (LS)* and like many routing algorithms in *MANETs* it is built upon a *MAC* data link layer.

Location Service. This service allows to access the node's geographical location via a location sensor such as a GPS or location beacons. The location service offers the following two primitives:

LS-GETPOSITION(): retrieves the node's current geographical location.
LS-GETDISTANCE(l_1, l_2): returns the distance between locations l_1 and l_2.

MAC layer. In *MANETs*, the *MAC* layer is used to broadcast and receive message in a node's neighbor. We model this communication capability via the two following primitives:

MAC-BROADCAST(m): sends the message m to all nodes located in the sender's neighborhood.
MAC-DELIVER(m): acts as a callback when message m is received.

2.2 Implementation

At the heart of *6SB* lay the 6SB-BROADCAST and the 6SB-DELIVER primitives presented previously, as well as two important functions, namely GETTARGETS, and GETDELAY. The detail of the *6SB* algorithm os presented in Algorithm 1.

The initialization phase. *6SB* uses three global variables: (1) the *threshold* indicates after how many received retransmissions, a message does not need to be forwarded anymore. (2) The $forwardZoneSize$ indicates the size of the forward zone, and (3) the message *counter* list, which is indexed by message IDs and keeps track of the number of received retransmissions (lines 2-5).

The broadcast primitive. When the 6SB-BROADCAST(m) primitive is called with a message m as parameter, a broadcast is initiated. This process is in charge of adding the six targets to m before it is broadcasted to the neighborhood via the MAC-BROADCAST primitive (lines 6-9). These targets are retrieved by the target computing function GETTARGETS.

The target computing function. Targets are computed via the GETTARGETS function by calculating the coordinates of each target based on the node's location and on its technical range (lines 22-31). The node's location is obtained via the location service's LS-GETPOSITION primitive.

The delivery primitive. When the *MAC* layer receives a message m for the first time through the MAC-DELIVER callback , the 6SB-DELIVER callback is triggered and the counter for m is set to 1. Then the node waits for a delay determined by the GETDELAY function if the delay is valid. During this waiting time, when other copies of m are received, they increment the message *counter*. After the waiting time elapses, m is forwarded using the 6SB-BROADCAST primitive if its *counter* is less or equal to the *threshold* (lines 10-21).

```
1: uses MAC, Location Service
2: INIT
3:    threshold ← ...                                   {threshold fixed by the user}
4:    noForwardZoneSize ← ...          {size of the no forward zone fixed by the user}
5:    counter ← ⟨0, 0, ..., 0⟩                          {list of counters}

6: To execute 6SB-BROADCAST(m) :
7:    m.targets ← GETTARGETS()                          {set the targets}
8:    m.center ← LS-GETPOSITION                         {set the current position}
9:    MAC-BROADCAST(m)                                  {broadcast the msg}

10: 6SB-DELIVER(m) occurs as follows:
11:    upon MAC-DELIVER(m) do                           {when MAC deliver a msg}
12:       if counter[m] = 0 then                        {if msg was already recieved}
13:          6SB-DELIVER(m)                             {delivers the msg, ¬ blocking}
14:          counter[m] ← 1                             {initial msg counter value}
15:          delay ← GETDELAY(m.targets, m.center)      {delay to wait}
16:          if delay ≥ 0 then                          {if the delay is valid}
17:             WAIT(delay)                             {wait until delay}
18:             if counter[m] ≤ threshold then          {if threshold is not reach}
19:                6SB-BROADCAST(m)                     {forwards m}
20:       else
21:          counter[m] ← counter[m] + 1                {increment the msg counter}

22: function GETTARGETS() :
23:    p ← LS-GETPOSITION                {get the current position of the node}
24:    p1 ← ⟨p.x + range, p.y⟩                          {computes the targets}
25:    p2 ← ⟨p.x − range, p.y⟩
26:    p3 ← ⟨p.x + sin(30) ∗ range, p.y + cos(30) ∗ range⟩
27:    p4 ← ⟨p.x + sin(30) ∗ range, p.y − cos(30) ∗ range⟩
28:    p5 ← ⟨p.x − sin(30) ∗ range, p.y + cos(30) ∗ range⟩
29:    p6 ← ⟨p.x − sin(30) ∗ range, p.y − cos(30) ∗ range⟩
30:    targets ← ⟨p1, p2, p3, p4, p5, p6⟩
31:    return targets                                   {returns the targets}

32: function GETDELAY(targets, center) :
33:    dist ← −1
34:    p ← LS-GETPOSITION                {get the current position of the node}
35:    for all t ∈ targets do                          {get the nearest target}
36:       if (LS-GETDISTANCE(p, t) < dist) ∨ (dist = −1) then
37:          dist ← LS-GETDISTANCE(p, t)
38:    if LS-GETDISTANCE(p, center) ≤ noForwardZoneSize then
39:       return −1                {if the node is in the no forward zone}
40:    else
41:       return DELAY(distance)          {return delay in function of distance}
```

Algorithm 1. Six-shot broadcast algorithm

The delay computing function. This function is central to the *6SB* algorithm. When GETDELAY is called with the targets contained in a message as parameters, the function first looks for the closest target. If the node is in the forwarding zone of the target, a delay is computed proportionally to the distance *dist* by the DELAY() function. We encapsulated this last function since it must be fine tuned according to the application context and the technology used. If the node is outside the forwarding then -1 is returned to indicate that no delay applies to the situation (lines 32-39).

3 Related work

Many different types of algorithms for broadcasting in *MANETs* exist. In the following, we investigate the algorithms closest to *6SB*. We first present the *counter-based scheme*, which created the foundation of the *wait and see* mechanism used in *6SB*. Then we investigate two context-aware algorithms, namely the *power-aware message propagation algorithm* and the *optimized flooding protocol*, which fine tune waiting delays according specific locations, similarly to *6SB*.

3.1 Counter-based scheme.

In the *counter-based scheme (CBS)*, when a node receives a new message m, it fixes a random waiting delay before making the forwarding decision. During this delay, the node counts the number of retransmission of m it receives. After the waiting delay has elapsed, the message is only forwarded if the number of retransmissions is smaller than a predetermined *threshold*. As a result, the forwarding ratio depends on the node density, i.e., in low density areas this ratio will be high, whereas in high density areas it will be very low. Simulation results presented in Section 4 show that this simple algorithm turns out to be very competitive as it does not concede much reliability for highly improved efficiency.

3.2 Power-aware message propagation algorithm.

The *power-aware message propagation algorithm (PAMPA)* has the exact same mechanism as *CBS* except that the delay is not fixed randomly but it is based on the intensity of the signal at which a message is received. The stronger the signal, the longer the delay. The idea behind *PAMPA* is that forwarders should be located as far away as possible from the source node. With its mechanisms, *PAMPA* ensures that only neighbors located on the outskirts of a node's neighborhood will forward messages.

3.3 Optimized flooding protocol.

The *optimized flooding protocol (OFP)* is a location-based broadcasting algorithm, which transposes the broadcasting problem in MANETs into the following geometric optimization problem: *how can we minimize the number of circles*

it takes to cover a certain surface under the constraint that every circle must have its center on the area covered by another circle? The answer to this problem is to divide the surface into hexagons the size of a circle, and then to draw a cercle on each summit. Transposed back to broadcasting in MANETs, the center of each cercle represents the ideal location for forwarders. Figure 2 depicts this idea with a central source node broadcasting a message, which will first be forwarded by three first-hop forwarders and then by six second-hop forwarders and so forth. This solution allows to have the absolute minimum number of forwarders in order to reach all nodes within a surface in an ideal case. The *OFP*

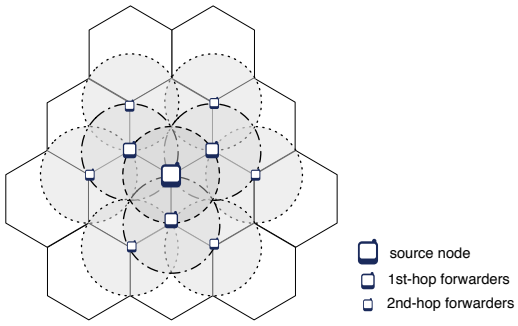

Fig. 2. OFP principle

algorithm works also like *CBS* except that the waiting time is set according to the distance of each node with the closest ideal forwarding spot. The closer to the spot, the shorter the waiting time.

3.4 Summary

All algorithms presented above use the *wait and see* mechanism introduce by *CBS*. *6SB*, *OFP* and *PAMPA* all add a context-based scheme in order to delegate the forwarding task to nodes located nearby strategic positions. Each of these algorithms defines these positions differently. *PAMPA* selects nodes located closest to the limit of its neighborhood, *OFP* selects nodes closest to three targets, and *6SB* selects node closest to six targets. Figure 3 illustrates each algorithm's forwarding pattern.

Targets are computed at different moments depending on the algorithm. In *6SB* targets are only computed by sending nodes before a message is sent. Other nodes only evaluate their distance to the targets associated with the message. In *OFP* on the other hand, every node computes the targets upon message reception, then it evaluates its distance to the closest target. With *PAMPA*, there are no defined targets and all receiving nodes evaluate their distance from the sender by measuring the signal strength at which messages are received.

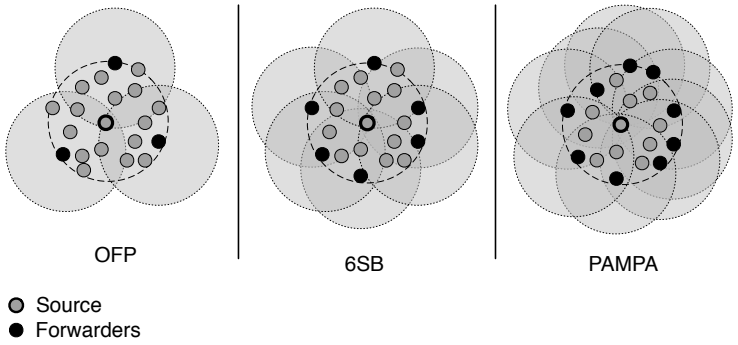

Fig. 3. forwarding patterns

4 Performance evaluation

We measure performance in terms of forward and delivery ratios, and compare them to the algorithms presented in the related work (Section 3). Our results change depending on the density of nodes in the field. In high densities, *6SB* performs as well as *OFP*, which is the best algorithm. In low densities, *6SB* performs better than *OFP* in term of reliability for small overhead in terms of efficiency. Our results also demonstrate that minor mobility has no influence on the performance results. Before discussing these results in detail, we present the simulation settings used for the evaluation.

4.1 Simulations environment

To compute the delivery and forward ratios of the algorithms, we ran simulations in the Sinalgo framework [8] written in Java. The context we simulate is one of an event, such as a conference or a festival, where every participant has a mobile device connected to a MANET with IEEE 802.11n WiFi links and a GPS chip. An example of such a device is Apple's iPhone or Nokia's E71. In our context, users are static or moving around at walking speed.

Density and connection degree. We define three parameters to characterize the density and the degree of connections of the network: the *map size*, the *technical range* and the *number of nodes*. We choose a square map of 200 meters width with a technical range of 20 meters. [1] In every simulation, one node broadcasts a message. Each simulation was run 100 times in an initially fully connected network of 160 up to 2000 nodes. This means that every node has an

[1] Note that IEEE 802.11n allows outdoor communications up to 140 meters between laptops and wireless routers and 70 meters for indoor communication. After testing the range of various smart phones, we found values between 20 and 30 meters to be the most reasonable.

from around 4 up to 60 neighbors. The map shape we use for the simulations is a torus to avoid having to deal with special conditions at the field limits. Table 1 summarizes the general simulations parameters we use.

Parameters	Value
Map size	$200\ m \cdot 200\ m$
Map shape	torus
Map area	$40000\ m^2$
Technical range	$20\ m$
Number of node	$160 - 2000$
Number of sender	1

Table 1. General simulations parameters

Mobility. There exist several mobility models and many studies about their realism in MANETs. One of the most used is the Random Waypoint Model [2]. Our torus-shaped map resolves a known issue of the Random Waypoint Model, i. e., the fact that node tend to converge to the center of the simulation field. In our context, we defined a scenario with mobility at walking speed. So, we set the speed of our Random Waypoint model uniformly distributed between 1 and 2 meters per second. In this mobility model, nodes choose a random waypoint on the field towards which they move. When the waypoint is reached, node wait for a random uniformly distributed random delay from 0 to 10 seconds before choosing another waypoint. The message transmission speed used in the simulation is 0.1 second and represents the time between a MAC-BROADCAST and a MAC-DELIVER in a 20 meter range. Table 2 summarizes the mobility and time parameters used.

Parameters	Value
Mobility	Random Waypoint in a torus
Delay	Uniform 0 - 10 s
Nodes speed	Uniform 1 - 2 m/s
Message transmission time	0.1 second

Table 2. Mobility parameters

4.2 6SB as good as OFP in high densities

Figure 4 and 5 present respectively the delivery and forward ratios of the *CBS*, *OFP*, *6SB* and *PAMPA* algorithms in a static setting *without mobility*.

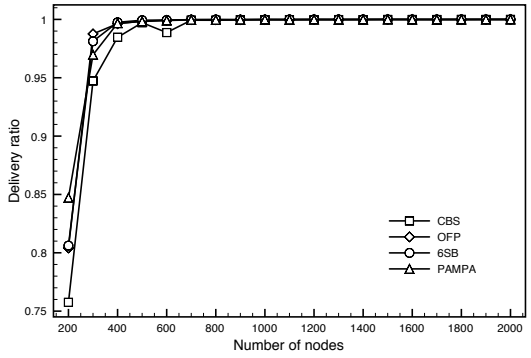

Fig. 4. Delivery ratio in high densities without mobility

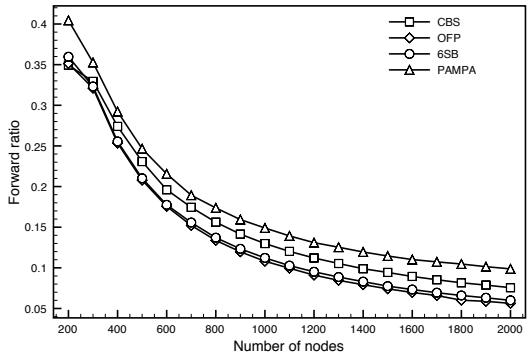

Fig. 5. Forward ratio in high densities without mobility

These results show that all evaluated algorithms perform well, between 5% to 40% of the nodes have to forward the messages. In high densities (more than 600 nodes), the different algorithms have no influence on the delivery ratio that is equal to 1, because there is enough nodes to propagate the message to the whole networks. The only difference is the forward ratio. The *OFP* algorithm is the one with the best forward ratio, closely followed by *6SB*. These two algorithms are clearly the best in terms of efficiency. *CBS*, despite its simplicity, turns out to be also very competitive. The *PAMPA* algorithm, which is more complicated, demonstrates poorer performances. Figure 6 and 7 present the same results in a mobile context. Mobility has almost no influence on the results. This is due to the significant difference between the slow walking speed at which nodes move and the high message propagation speed.

These results convey the fact that in high densities, when nodes have around 18 or more neighbors (above 600 nodes in our simulation settings), all evaluated

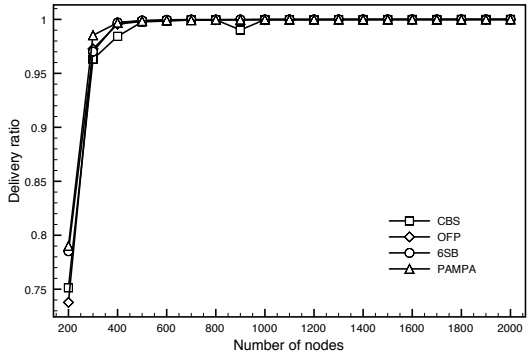

Fig. 6. Delivery ratio in high densities with mobility

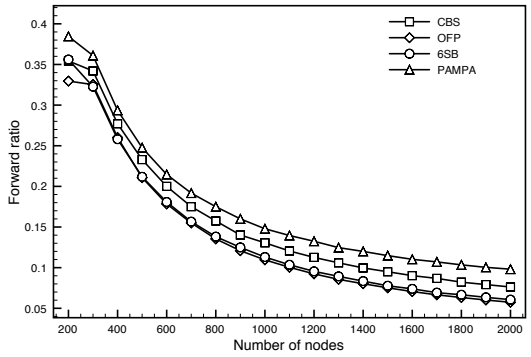

Fig. 7. Forward ratio in high densities with mobility

algorithms are reliable and *6SB* performs as well as *OFP*, which is the best algorithm in terms of efficiency. Mobility has no influences on the results.

4.3 6SB better than OFP in low densities

Figure 8 and 9 present respectively the delivery and forward ratios of *CBS*, *OFP*, *6SB* and *PAMPA* in low densities (until 350 nodes). *PAMPA* and *6SB* have the highest delivery ratios. *CBS* has a delivery ratio, which is always the lowest, but it tend to join *PAMPA* and *6SB* in higher densities. *OFP* has an intermediate delivery ratio. It begins at the same low value than *CBS*, but becomes reliable more rapidly. In terms of efficiency, *OFP* has always the lowest forward ratio. *CBS* has a forward ratio a little bit higher, followed then by *6SB* and *PAMPA* in the last position. When the density increases, these ratios tend to change; *6SB* became more efficient than *CBS*.

These results show clearly that there is a tradeoff between reliability and efficiency in lower densities. *PAMPA* and *6SB* focus on reliability, since they have a higher delivery ratio for a higher forward ratio. *OFP* and *CBS* focus on efficiency, with a lower delivery ratio for a lower forward ratio. To measure this tradeoff, we have computed the number of delivers gained with one more forward in comparison to OFP, which is the best algorithm in term of forward ratio (see Figure 10). This is the *utility* of having one more forwarder. *CBS* is the worst algorithm because it has the lowest gain for the higher variability of the results. *PAMPA* has a small utility just over 0. *6SB* has the best utility with values above 1. This means that for one more forward, *6SB* will generate more than one extra deliver. So *6SB* allows to have a better delivery ratio than *OFP* for the lowest overhead in term of forward. *6SB* offers the best tradeoff between reliability and efficiency in lower densities.

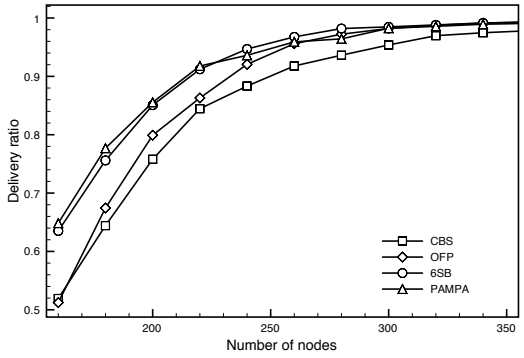

Fig. 8. Delivery ratio in low densities

5 Concluding remarks

Finding the right algorithm for broadcasting in a MANET is not a trivial task. The problem of broadcasting in MANETs is a tradeoff between reliability and efficiency. Reliability is measured by the number of nodes that deliver the message and efficiency by the number of nodes that forward the message. A maximum of nodes have to deliver the message, while we try to minimize the number of nodes that forward this message. By optimizing the number of forwarders, we save resources, such as battery power, but we increase the risk that some nodes do not receive the message.

The main contribution of this paper is *six-shot broadcast* (*6SB*), a context-aware communication algorithm using location information to fine-tune its forwarding process. Our extensive performance evaluations show that *6SB* com-

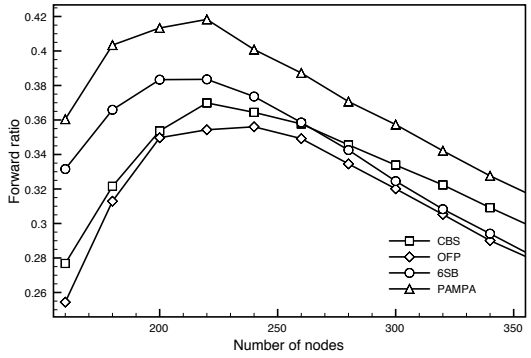

Fig. 9. Forward ratio in low densities

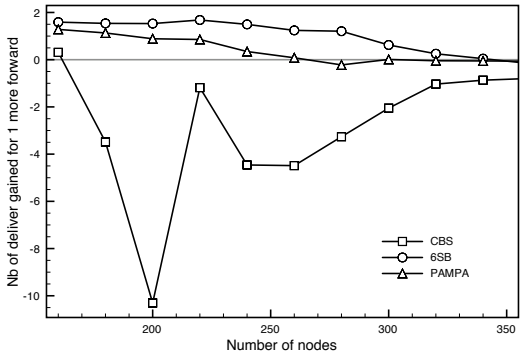

Fig. 10. Number of deliver gained for one more forward in comparison to OFP

petes with the most efficient algorithms in high densities and offers increased reliability at a reasonable price in low densities.

We intend to extend this research in two directions, first we will have a look at a practical implementation of the *6SB* algorithm and its use on mobile devices for a concrete mobile application. Second, we will further examine *6SB* theoretically by conducting simulations in different contexts and by integrating the *6SB* algorithm into other broadcast-based algorithms, such as *FIFO-broadcast*, or *atomic-commit*.

References

1. Reka Albert and Albert-Laszlo Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.
2. T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue*

on Mobile Ad Hoc Networking: Research, Trends and Applications, 2(5):483–502, 2002.

3. Patrick Th. Eugster, Benoît Garbinato, and Adrian Holzer. Design and implementation of the pervaho middleware for mobile context-aware applications. In *Proc. of MCETECH'08*, 2008.
4. Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999. ACM.
5. Qing Huang, Yong Bai, and Lan Chen. Efficient lightweight broadcasting protocols for multi-hop ad hoc networks. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium*, pages 1 – 5, 2006.
6. Hugo Miranda, Simone Leggio, Luis Rodrigues, and Kimmo Raatikainen. Removing probabilities to improve efficiency in broadcast algorithms. In *Proceedings of the 5th MiNEMA Workshop, Middleware for Network Eccentric and Mobile Applications*, pages 20–24, 2007.
7. Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, New York, NY, USA, 1999. ACM.
8. Distributed Computing Group of ETH Zurich. Sinalgo: a simulation framework for manets. *http://dcg.ethz.ch/projects/sinalgo/*.
9. Francisco Javier Ovalle-Martnez, Amiya Nayak, Ivan Stojmenovic, Jean Carle, and David Simplot-Ryl. Area-based beaconless reliable broadcasting in sensor networks. *IJSNet*, 1(1/2):20–33, 2006.
10. Vamsi Paruchuri, Arjan Durresi, and Raj Jain. Optimized flooding protocol for ad hoc networks. *CoRR*, cs.NI/0311013, 2003. informal publication.
11. B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 194–205, 2002.