Message Stability and Reliable Broadcasts in Mobile Ad-Hoc Networks

Kulpreet Singh, Andronikos Nedos, Gregor Gärtner, and Siobhán Clarke

Distributed Systems Group, Trinity College, Dublin, Ireland

Abstract. Many to many reliable broadcast is useful while building distributed services like group membership and agreement in a MANET. Efforts in implementing reliable broadcast optimised for MANETs have resulted in new protocols that reduce the number of transmissions required to achieve reliable broadcast. A practical implementation of reliable broadcasts requires the ability to detect message stability, and there is still a need to develop protocols that efficiently support message stability determination in a MANET. In this paper we describe such a protocol that is independent of the broadcast optimisation being used, and focuses on providing efficient message stability. As the main idea of the protocol, we define a message dependency relationship and use this relationship to implement reliable broadcast as well as message stability detection. Simulations for mobile and static scenarios show our protocol has only a minimal performance degradation with node mobility.

Keywords: message stability, reliable broadcast, MANET

1 Introduction

In mobile ad-hoc networks (MANETs) frequent topology changes and changing membership pose new challenges for application developers. Distributed services like replication, agreement and group membership help developers build applications for MANETs. Many to many reliable broadcast is useful while building these distributed services.

A practical implementation of many to many reliable broadcast service requires addressing the problem of determining when a message has become stable at the participating nodes. A message is said to be stable when all nodes participating in the reliable broadcast have received the message; guaranteeing that no node will ask for the message again. This allows nodes to delete the stable message from their buffers. In this paper we describe a many to many reliable broadcast protocol that supports efficient determination of message stability in a MANET, and show how it scales with node mobility and node density.

The main idea behind the protocol is the use of dependency relationship between messages broadcast over multiple hops and using these relationships to implement reliability and determine message stability. In this paper we describe this dependency relationship in detail and show how this relationship is used to implement reliable broadcast and further how it allows each participating node to

V.R. Sirotiuk and E. Chávez (Eds.): ADHOC-NOW 2005, LNCS 3738, pp. 297–310, 2005.

determine independently whether a message has been received and delivered at all other nodes. Finally, we show how this ability is easily extended to determine message stability.

In the course of developing reliable broadcast protocols for MANETs, numerous broadcast optimisations have been developed. These optimisations are required to handle the broadcast storm problem as first presented in [1]. The same paper also presented some initial solutions to the problem. The protocol described in this paper is architectured so that it allows any of the optimisations to be deployed below our broadcast and the stability determination algorithm. This allows for flexible use of our scheme, depending on the optimisation available in the MANET.

Given the exorbitant energy costs of message transmissions [2] in a MANET, we trade reduced message transmissions for increased computation for every message stabilised.We show how we achieve the same by using implicit acknowledgements through message dependencies. This requires working with large complex data structures, but saves on the number of message transmissions.

The protocol we present can be easily extended to provide an agreement protocol and further a virtually synchronous membership service, but we do not describe these in this paper. The theme of this paper remains achieving message stability in a MANET. Simulation results show that our protocol is not adversely affected by node mobility; and the number of message transmissions required to determine message stability is dependent only on the size of the network.

2 Related Work

Given the importance of reliable broadcast for building applications for MANETs, and recognising the challenges presented by the shared medium in a MANET, a lot of effort has been focused towards reducing redundant broadcasts and improving the reliability of broadcasts in a MANET.

A redundant broadcast is called redundant as it does not communicate a broadcast message to any additional nodes. Numerous techniques to reduce redundant broadcasts and improve reliability have been developed. These techniques vary in the amount of information utilised to recognise a broadcast as redundant or not. The simplest techniques are probabilistic flooding as presented in [3][4] and the counter based technique presented in the seminal paper identifying the broadcast storm problem [1]. Other techniques as [5][6][7][8] involve keeping track of neighbourhood information and possibly the path that a message took to the current node. Other techniques such as [9] utilise the information available from the underlying routing protocol.

Work has also gone into comparing these techniques through different simulations parameters. [10] and [11] present detailed study of how various techniques compare against each other. [11] also presents a clean characterisation of various techniques depending on the knowledge utilised by a node to determine whether it should rebroadcast a received message.

For practical implementations of reliable broadcasts and other distributed services it is imperative to efficiently determine message stability. To our knowledge [12] is the only system implementation presented in literature that addresses the problem of message stability detection in MANETs. The system uses a gossip based approach to detect message stability. The gossipped messages act as explicit acknowledgements for messages, in contrast, our system uses implicit acknowledgements implemented via message dependencies. Baldoni et al. in [13] discuss a session based approach to implement message stability in partitionable MANETs.

3 Reliable Broadcast and Message Stability

In this section we present in detail our approach to implementing a reliable many to many broadcast that allows efficient determination of message stability. We first describe our approach towards using an optimised broadcast to implement a many to many reliable broadcast. Then we show how this reliable broadcast is used to determine when a message has been received and delivered at another node. Finally, we show how this knowledge is collected to determine message stability.

3.1 Implementing Reliable Broadcasts

As mentioned earlier, we allow any broadcast optimisation to be utilised to implement our reliable broadcast and thus the message stability protocol. For the purposes of this paper we describe the use of a counter based scheme [1] to reduce redundant broadcasts; the same scheme is used in our simulation experiments as well.

Reliability is implemented using negative acknowledgements (nacks). A transmitting node assumes that the messages are received at its neighbours. This results in an optimistic approach for sending messages, wherein a node keeps transmitting messages assuming the earlier messages have been received by one or more of its neighbours. When a neighbour of the sending node realises it has missed a reception, it sends a nack for the missing message.

Apart from the information in message headers required to implement the broadcast optimisation we provide additional information in the headers to implement reliability and determine message stability. This extra information includes the message identifier (*mid*) of the message being sent. A message identifier is composed of the *id* of the sending node and the *sequence number* of the message; thus p_i is a message sent by p with sequence number i.

Before we describe the message dependency relationship in detail and the working of our protocol, we provide some terminology. We say a message is *sent* by a node if the node originates the message and is the first node to transmit the message. A message is said to be $forwarded$ by a node if it receives a message sent by some other node and then retransmits it. A message is said to be *delivered* at a node if it has been received by the node and after fulfilling certain

conditions is handed to the application at the node. The conditions required to be satisfied before a message is delivered are described in Section 3.3. The next section now describes message dependencies.

3.2 Message Dependencies

A message p_i is said to be dependent on a message q_j if p receives and delivers q_j before sending p_i or if $q = p \wedge j = i - 1$. The dependency relationship is transitive and is described in detail next.

If p sends a message p_i, p_{i-1} is a dependency of p_i, we call p_{i-1} the *last sent* dependency of p_i. A reliability scheme based only on the *last sent* dependency will work fine, but since we know a sending node has also received and delivered other messages from other nodes, we compound the message header with another dependency. This allows a receiving node to send a nack for other messages that the sending node delivered before sending p_i.

The other dependency is the message last delivered by the sending node; that is if a node p receives and delivers a message q_k and then the first message it sends is p_i, it includes a dependency for q_k in the header for p_i. This dependency is called the *last delivered* dependency of p_i. Our protocol, as described in Section 3.3, requires that a node that receives p_i delivers both *last sent* and *last delivered* dependencies of p_i before delivering p_i.

The dependency relationship is transitive and thus allows nodes to work towards synchronising their progress on message exchanges. Since all participating nodes act as both senders and receivers, the dependency relationship is used to implicitly gather acknowledgements. In the next section we show how reliability is implemented using the dependency relationship.

3.3 Reliability via Dependencies

If a message m is a dependency of n, we write $m \rightarrow n$. If $m \rightarrow n \wedge n \rightarrow n_1 \ldots n_l \rightarrow o$ then m is called the *eventual dependency* of o, written as $m \triangleright o$. When we write a single letter like m for a message, it signifies that the sender of the message is not important in the current context.

We now state the two fundamental rules of our protocol —

$\mathcal{R}$1 A message received by a node is not delivered until all its dependencies have been received and delivered at the receiving node.

$\mathcal{R}$2 A message is not forwarded by a node till it has been delivered at that node.

From the definitions and the rules described till now, we state the lemmas —

Lemma 1. *A message m delivered by a node p, is an eventual dependency of any message, p_k, sent by p after delivering m.*

Proof. From the definition of dependencies, the first message p_i sent by p after delivering m includes m as a dependency; by the definition of eventual dependency, all messages p_j sent by p such that $j > i$ will have m as an eventual dependency.

Lemma 2. *A node p delivers a received message n only if all messages m such that $m \triangleright n$ have been delivered by p.*

Proof. Follows from the definition of eventual dependency and the rule $\mathcal{R}1$. □

Corollary 3. *If a node p delivers a message n it has delivered all messages m such that $m \triangleright n$.*

Negative Acknowledgements. Apart from the rules $\mathcal{R}1$ and $\mathcal{R}2$, we now describe how negative acknowledgements are generated when a node receives a message while some of the message's dependencies have not been delivered at the receiving node. Suppose a node p receives a message q_i from q with two dependencies as q_{i-1} and s_j. From rule $\mathcal{R}1$, p will deliver the message only if p has received and delivered both q_{i-1} and s_j.

If p can not deliver q_k, it is because some eventual dependency of q_k has not been received by p. This requires p to nack for some eventual dependency of q_k. To find which eventual dependency to nack for, p checks if it has received either of the two dependencies of q_k. If it has not received any one of two dependencies, p nacks for the missing dependency. If it has received the dependencies, p recursively checks if the dependencies of the already received dependencies have been received. When a dependency is found which has not yet been received at p, p sends a nack for the same. We now state the third rule for our protocol.

$\mathcal{R}3$ For every message received that cannot be immediately delivered, the receiving node sends a nack for some eventual dependency of the received message which has not yet been received.

Our system requires nodes to send messages at regular intervals to allow for nodes to receive messages which help them determine missing dependencies. This regular timeout messages sent by each node makes our system best suited for applications that require all-to-all streams of regular messages, like group membership, replication and consistency management.

Reliability Proofs. In this section we present the theorems for reliability and arguments for their correctness; but first we define the concept of reachability. Two nodes p and q are said to be *reachable* from each other if they can send and receive messages to and from each other, otherwise they are said to be *unreachable*. Nodes can become unreachable from each other due to mobility related link failures or processor failures. As an example, consider nodes p and q being able to send and receive messages through an intermediate node r. If r failstops and if there are no other intermediate nodes that forward their messages, then p and q become unreachable from each other.

Further, we assume for the sake of our correctness arguments that we do not run into the fairness problem [14] encountered with the IEEE 802.11 MAC protocol. The eventual fair broadcast assumption guarantees that every participating node broadcasts and receives messages and is stated as —

Eventual fair broadcast assumption — In an infinite execution, each node broadcasts messages infinitely often, the immediate neighbours of the node receive infinitely many of those messages and, if a message is re-broadcast infinitely often, all neighbours of the sending node eventually receive that message.

To allow a broadcast message to be propagated to the network we introduce the final rule for our protocol.

$\mathcal{R}4$ Every message that is received and delivered by a node is forwarded by the receiving node.

Rules $\mathcal{R}2$ and $\mathcal{R}4$ work together to first make sure a message is not forwarded till it has been delivered, and at the same time a message is guaranteed to be forwarded when it has been delivered. From the assumptions and rules described, we now state the property that guarantees the dissemination of a message.

Eventual Reception Property — For any message, if any node p broadcasts or receives a message, then every node that remains reachable from p, eventually receives it.

To establish this property we prove Theorems 4, 5 and 6. But before we do so, we state one final assumption —

No spurious messages assumption — All messages received by any node have been sent by some participating node.

Theorem 4. *Given two nodes p and q remain neighbours, if q receives a message m forwarded by p, q eventually delivers the message.*

Proof. From Corollary 3, p has received and delivered all eventual dependencies of m, and from rule $\mathcal{R}3$, q nacks for missing dependencies of m. Given p and q remain neighbours and the eventual broadcast assumption, q eventually delivers m. □

Theorem 5. *Every message sent by a node p is eventually received by every node that remains reachable from p.*

Proof. The proof for this theorem is presented in Appendix I.

Theorem 6. *Every message received by a node p is eventually received by every node that remains reachable from p.*

Proof. When p receives the message m, by Theorem 4, p delivers m; by rule $\mathcal{R}4$, p forwards m; from the no spurious messages assumption m has been broadcast by some participating node and finally from Theorem 5 all nodes receive m. □

The dependencies and the many to many reliable broadcast allow us to implement an useful facility called the "Observable Predicate for Deliver", or OPD. The OPD allows a node to determine if another node has yet received and delivered a message sent by any of the participating nodes. The next subsection describes how we implement the OPD and then we describe how the OPD is used to determine message stability.

3.4 Observable Predicate for Delivery

The *Observable Predicate for Delivery*, or OPD, allows a node p to determine if the sender of a certain message q_i has received and delivered another message r_j before sending q_i. We say, $OPD(p, r_j, q_i)$ is true if node p can determine that the node q has delivered the message r_j before q sent q_i. The Trans broadcast system in [15] first employed the property to build a partial relation between messages and to determine message stability.

For the purposes of evaluating the OPD the protocol maintains a directed graph at each node. The directed graph is a graph of message identifiers. We call this directed graph the "Delivered Before Graph", or DBG. A message identifier for message s_k sent by s has two edges to it in the DBG, one from each of s_k's dependencies. Thus there are edges in to s_k from s_{k-1} and the message last delivered at s when s_k was sent.

The main idea for determining the OPD is due to transitivity relationship in message dependencies and the way DBG is constructed. In a DBG there is a path from a message m to n if and only if $m \triangleright n$. Also from rule $\mathcal{R}2$, if $m \triangleright n$, we know that the sender of n is guaranteed to have delivered m before sending n. We later prove these statements as Theorem 7, but first we state the conditions under which $OPD(p, r_j, q_i)$ evaluates to true. The $OPD(p, r_j, q_i)$ is true if —

1. p has delivered a sequence of messages starting from message r_j and ending in message q_i.
2. There is a path from r_j to q_i in p's DBG.

Theorem 7. *If there is a path from message p_i to q_j in a node r's DBG, then q has received and delivered p_i before broadcasting q_j.*

Proof. From the the definition of DBG, $p_i \triangleright q_j$; from rule $\mathcal{R}2$ q has delivered q_j; finally from Corollary 3, q has delivered p_i. □

3.5 Message Stability

In this section we describe how message stability is determined using OPD. We assume the membership of the mobile ad-hoc network under consideration is known to be the set of nodes, $\mathcal{M}$, and remains the same during our system runs. We take the liberty of this assumption for this paper, but our broadcast scheme allows tracking of virtual synchronous membership as elaborated further in [16]. For the simulations, we start measuring message stability only after an initial membership has been installed. Since we are interested only in measuring message stability characteristics, we chose scenarios where there are no partitions and no nodes fail.

Given the OPD can be evaluated at all the participating nodes, determining message stability becomes straight forward. On reception of a message, the receiving node uses the OPD mechanism to evaluate if any of the unstable messages has now become stable.

To determine the same we introduce the "Message Last Delivered" *mld* array. This array maintains references to the messages last delivered at each node from all other nodes in $\mathcal{M}$. Thus, $mld[p]$ at node q is the message last delivered from p at node q. Given the definitions of *mld* and OPD we have the following lemma —

Lemma 8. *A message q_i is determined as stable at node p when $\forall n \in \mathcal{M}$ $OPD(p, q_i, mld[n]) = true$.*

Proof. The proof follows from the definition of OPD. When the condition in the lemma is true, p knows that the message q_i has been delivered at all nodes participating in the broadcast, and thus q_i is stable at p. □

Next we prove the liveness of the protocol for determining message stability. We state the theorem —

Theorem 9. *In an infinite execution of the protocol, when nodes do not become unreachable from each other, a message q_i eventually becomes stable at all participating nodes.*

Proof. From Theorem 6, in an infinite execution the message q_i is eventually received at all participating nodes. Further from the eventual broadcast assumption, all participating nodes will broadcast messages which by Lemma 1 will have q_i as an eventual dependency. Again from Theorem 6, these messages will be received at all other nodes, thus resulting in q_i being determined stable at all participating nodes. □

Our message stability approach involves checking for paths in the DBG. This is the computationally intensive; one that we consider as a good trade off for reduction in number of message transmissions.

4 Simulations

The goal of the simulation experiments is to verify that the OPD based mechanism for determining message stability scales well with the number of nodes in the network, the number of nodes per unit area (node density) and node speeds.

We use the ns2 simulator (v 2.27) [17] with the two-ray ground model and a transmission range of 250m, using the random way point mobility model. The number of nodes vary from 4 to 36, and simulations are run for both static and mobile scenarios. For the mobile scenarios we compare runs using speeds of 2 m/s and 8 m/s. Each of the scenarios are run 10 times and all values shown are an average of the results generated from the scenarios. We do not run simulations for more than 36 nodes as it runs into scalability issues with ns2 when all-to-all broadcasts are simulated.

To observe the effects of network size and node density we ran simulations with static nodes while varying the distance between the nodes. We use a square grid of n x n nodes and vary the distance between nodes along the length and

the breadth of the grid. We use 40, 100, 140 and 200 meters as the inter-node distance, d. The maximum distance between nodes is thus the distance between diagonally opposite nodes, viz $\sqrt{2}(n-1) * d$. So for the 36 node, 40m scenario, the maximum distance between nodes is 283m. With the transmission radius set to 250m, a single transmission from a node covers almost all nodes in the most dense and largest network. Whereas for the least dense network, with $d = 200m$, the maximum distance between nodes is about 1414m, requiring a number of forwards of a message to propagate the message across the network.

To isolate the effects of node density from mobility scenarios we chose the mobility scenarios such that the ratio of the aggregate area covered by the radio ranges of all nodes to the geographical area is the same. We call this ratio the "coverage ratio". For the mobility scenarios we kept this ratio equal to 10. Thus given the number of nodes and the transmission radius, we can determine the geographical area to run the simulations.

To compare the mobile cases to the static case, we ran simulations for a static network with the coverage ratio same as that chosen for the mobile network. With inter-node distance, $d = 140m$, we get the coverage ratio of 10 in the static scenarios, the same as that for the mobile scenarios. In the figures below we have plotted this static case both in the graph for static and mobile network, this helps us observe the affect of mobility, given networks with same node density.

Timeouts. For the simulations each node sends a message at variable time intervals; all participating nodes act as both senders and receivers. The time interval between message transmissions by a node is made proportional to the number of neighbours of the sending node. Each node starts by using a randomly selected time period between 0.1 and $0.15sec$. Given the number of neighbours[1], $|neighbours|$, a sending node then chooses a random timeout interval from between $0.1 \times |neighbours|$ and $0.15 \times |neighbours|$. Such an approach results in timeout intervals of up to 4 sec in a high density network. The timeout approach described above reduces the probability of collisions, but increases end to end latency of communication.

4.1 Results

All figures present two graphs, one for the static network and the second for the mobility scenario. The graph for the mobility scenarios includes one curve from the static scenarios with the same coverage ratio. This way we are also able to compare the performance of the mobile scenario with static scenarios with different node density.

One of the metrics we use is the number of messages transmitted in the entire network while stabilising a message. To measure the same, we count the average number of message transmissions in the entire network after a message is sent, till the message is stabilised at all nodes. To capture the latency aspect

[1] A node easily determines the set of its neighbours by looking at the messages received and whether they were forwarded to reach this node or not.

of our system's performance we later present results for time required to deliver messages across participating nodes.

Figure 1 shows the number of messages required to determine message stability. For the static cases we see a linear increase in the number of transmissions with the increase in the number of nodes in the network. We also see that the number of message transmissions required does not show much variation with the density of the network. This comes as a surprise to us, as we expected high density network to require fewer retransmissions. This could be a result of either the counter based scheme not saving too many rebroadcasts or the minimal broadcast optimisations provided by a counter based scheme being counteracted by the increase in number of collisions for dense networks.

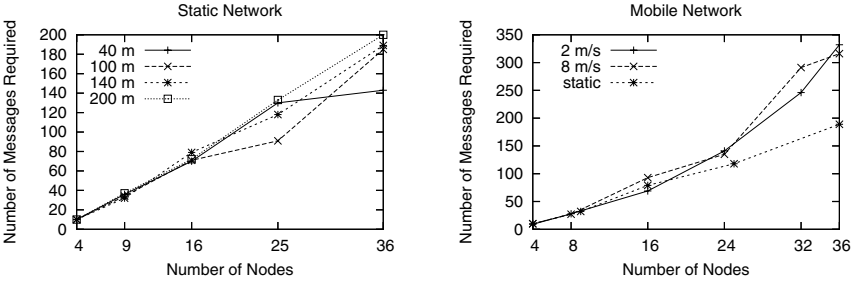

Fig. 1. Average number of messages required to determine stability

The mobile scenarios show an interesting behaviour as well. We see a similar performance up till about 25 nodes for both the mobile cases and the static cases. But at 32 nodes, the number of transmissions required for both the mobile scenario are similar and only about 50% higher than those for the static case. This shows our protocol handles mobility quite well.

Figure 2 shows the average number of times each message is forwarded before it stabilises on all the participating nodes. Again we see a linear increase over the

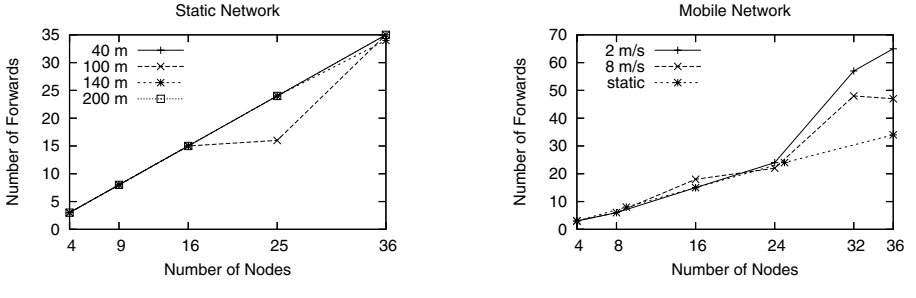

Fig. 2. Average number of times a message is forwarded

static scenarios and no change with the density of the network. The interesting result is again when we compare the mobile and static scenarios with similar node density. The mobile and the static cases show similar performance till about 25 nodes, and the mobile cases then show a 50% to a 100% increase in the number of times a message is forwarded as compared to the static cases. We also see the $8m/s$ scenario showing a better performance than the $2m/s$ scenario, this is probably because of spatial reuse [18] caused by higher mobility.

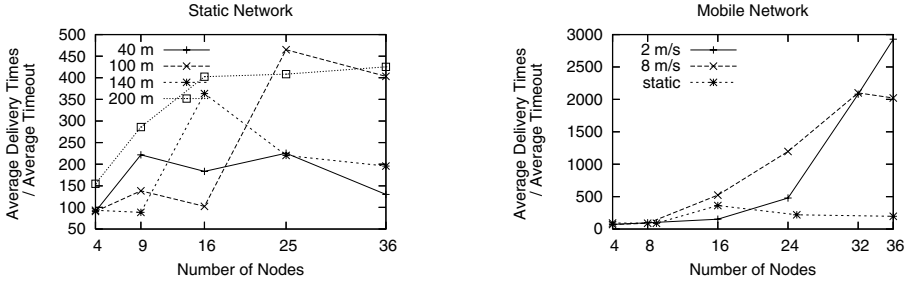

Fig. 3. Average Delivery Times

Figure 3 shows how the message delivery time is affected. To measure the delivery time we measure the average time elapsed between a message send and the times the message is delivered at all nodes. We then take the ratio of this average delivery time of a scenario and the average timeout used by all nodes in that scenario. This way we normalise the effect of the timeouts used for different scenarios. In other words, the graphs show the delivery times for the scenarios if the timeout was 1 *sec*.

The results for the static case show a wide variation over the node density, yet there is a tendency for the delivery times to normalise as the number of nodes increases. The graphs shows that the delivery times are affected by the node density. An interesting observation is that the normalised delivery time for the static scenarios are always between 100 and 450 *msec*. For the mobile cases the delivery times are close to those of the static scenarios till about 25 nodes, but are substantially higher for more than 25 nodes. This shows that for a large network, mobility does affect the delivery times of a message; even while the effort required to stabilise a message is not affected that much.

5 Conclusions

In this paper we presented an approach to providing a reliable broadcast using message dependencies. The approach does not require explicit acknowledgements, instead message dependencies provide implicit acknowledgements. The reliable broadcast allows the use of any existing broadcast redundancy reduction techniques. We further presented how this reliable broadcast which uses

message dependencies can be used to determine message stability, an important property for practical implementations of a broadcast protocol.

Our simulation results show that our protocol scales well with node density of the network and node mobility. In fact for networks with up to 25 nodes the results for static and mobile cases are very similar. The simulation results give us confidence that our broadcast scheme should be used to implement higher level services like group membership and agreement. We are implementing our protocol that will allow us to compare it with the one presented in [12].

References

1. Tseng, Y.C., Ni, S.Y., Chen, Y.S., Sheu, J.P.: The broadcast storm problem in a mobile ad hoc network. Wirel. Netw. **8** (2002) 153–167
2. Liang, W.: Constructing minimum-energy broadcast trees in wireless ad hoc networks. In: MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, New York, NY, USA, ACM Press (2002) 112–122
3. Sasson, Y., Cavin, D., Schiper, A.: Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In: Proceedings of IEEE Wireless Communications and Networking Conference. Volume 2. (2003) 1124 – 1130
4. Eugster, P.T., Guerraoui, R., Handurukande, S.B., Kouznetsov, P., Kermarrec, A.M.: Lightweight probabilistic broadcast. ACM Trans. Comput. Syst. **21** (2003) 341–374
5. Wu, J., Dai, F.: Broadcasting in ad hoc networks based on self-pruning. In: Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. Volume 3., IEEE (2003) 2240 – 2250
6. Lou, W., Wu, J.: On reducing broadcast redundancy in ad hoc wireless networks. IEEE Transactions on Mobile Computing **1** (2002) 111– 122
7. Wei, P., Xicheng, L.: Ahbp: An efficient broadcast protocol for mobile ad hoc networks. Journal of Computer Science and Technology (2001)
8. Sun, M., Feng, W., Lai, T.H.: Location aided broadcast in wireless ad hoc networks. In: Proceedings of IEEE Global Telecommunications Conference. Volume 5. (2001) 2842 – 2846
9. Jo, J., Eugster, P.T., Hubaux, J.: Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In: Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. Volume 3. (2003) 2229 – 2239
10. Kunz, T.: Multicasting in mobile ad-hoc networks: achieving high packet delivery ratios. In: Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research, IBM Press (2003) 156–170
11. Williams, B., Camp, T.: Comparison of broadcasting techniques for mobile ad hoc networks. In: MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, New York, NY, USA, ACM Press (2002) 194–205
12. Friedmann, R., Tcharny, G.: Stability detection in mobile ad-hoc networks. Technical report, Israel Institute of Technology (2003)
13. R.Baldoni, Ciuffoletti, A., Marchetti, C.: A message stability tracking protocol for mobile ad-hoc networks. In: 5th Workshop on Distributed Systems: Algorithms, Architectures and Languages (WSDAAL 2000), Ischia (Naples), Italy (2000) 18–20

14. Barrett, C.L., Marathe, M.V., Engelhart, D.C., Sivasubramaniam, A.: Analyzing the short-term fairness of ieee 802.11 in wireless multi-hop radio networks. 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02) (2002) 137 – 144
15. Moser, L.E., Melliar-Smith, P., Agarwal, V.: Processor membership in asynchronous distributed systems. IEEE Transaction on Parallel and Distributed Systems **5** (1994) 459–473
16. Singh, K.: Towards virtual synchrony in manets. Fifth European Dependable Computing Conference - Student Forum (2005)
17. Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y., Yu, H.: Advances in network simulation. IEEE Computer **33** (2000) 59–67
18. Guo, X., Roy, S., Conner, W.: Spatial reuse in wireless ad-hoc networks. In: IEEE 58th Vehicular Technology Conference. Volume 3. (2003) 1437–1442

Appendix I

Here we present the proof for Theorem 5. To prove the theorem we consider the four cases as presented in Figure 4. The cases (i) and (ii) assume static nodes and we use induction to prove the theorem under the static nodes assumption. Cases (iii) and (iv) allow for mobile nodes and we again use an induction to prove the theorem.

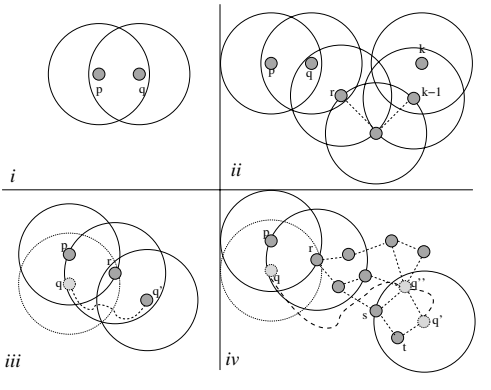

Fig. 4. Node p sends a message, q eventually receives the message

If a node p sends a message p_i, nodes that stay in the immediate neighbourhood of p eventually receive p_i. This is the case in Figure 4 (i).

case (i) From the eventual fair broadcast assumption, q eventually receives some message p_k from p, such that $k > i$. From Lemma 1, $p_i \triangleright p_k$, thus from rule $\mathcal{R}3$, q nacks for some dependency of p_k. By the definition of dependencies, q eventually nacks for p_i and then again from the eventual fair broadcast assumption, receives the same.

Next we prove under the assumption of static nodes, that if a node p sends a message, all nodes that remain reachable from p receive the message. The proof is an induction on the series of neighbouring nodes that a message goes through.

case (ii) Figure 4 (ii) presents this case. If the node p sends a message p_i, from case (i), q eventually receives the message, this is the base case for the induction. As the inductive step, we assume $(k-1)^{th}$ neighbour receives the message and prove that whenever $(k-1)^{th}$ neighbour receives p_i, $(k)^{th}$ neighbour receives p_i. Assume that the $(k-1)^{th}$ and k^{th} neighbours have identities $k-1$ and k, as shown in the figure. Then from Theorem 4 and rule $\mathcal{R}4$, $k-1$ delivers and forwards p_i. By Lemma 1 and the argument for case (i), k eventually receives p_i via k.

Next we consider mobility and show that if a node broadcasts a message all nodes that remain reachable from p receive the message, even if they are mobile and do not stay in the immediate neighbourhood of the sending node. First we prove that if a node q moves away to q' shown in Figure 4 (iii), q eventually receives the message sent by p.

case (iii) By case (i), r receives p_i and by Theorem 4, r delivers p_i. Then by $\mathcal{R}4$, r forwards p_i. This case then reduces to case (i).

Finally, we prove that if q moves such that there is a sequence of nodes that have to forward p_i, as in Figure 4 (iv), q still eventually receives the message. The proof for this case is an induction on the series of nodes the message is forwarded through to reach q.

case (iv) Case (iii) is the base case for the induction; the length of the series of intermediate nodes is 1. For the inductive step, we assume that if q moves to location shown as q'' as shown in Figure 4 (iv), and r is replaced by a series of nodes of length $|l-1|$, such that the last node in the series is s, q still receives p_i. The proof for the inductive step when the length of the series is l, and q moves to q' such that the last node in the series is t is shown now. Given the assumption for the series with length $l-1$, s has received p_i. From Theorem 4 and rule $\mathcal{R}4$, s delivers and forwards p_i. Again from the argument in case (i), t receives and delivers p_i via s and then q' receives p_i via t. □