Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

ELSEVIER

Computer Networks 42 (2003) 101–118

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Reducing the energy consumption caused by flooding messages in mobile ad hoc networks

S.Y. Wang

Department of Computer Science and Information Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan

Received 19 July 2002; received in revised form 3 December 2002; accepted 3 January 2003

Responsible Editor: S. Fdida

Abstract

The message flooding method is an important and useful operation for several applications. However, if a network has a large number of mobile nodes and these nodes use the flood operation often, a mobile node's scarce battery energy will be quickly consumed due to flooding too many messages.

In this paper, we propose an approach to reducing the energy consumption caused by flooding messages. The energy saving is achieved by (1) merging several small flood messages into a larger one, and (2) limiting the scope of a flood message. Our simulation results show that, at the cost of an increased message forwarding delay, the proposed approach can substantially reduce flood messages' energy consumption without increasing their delivery failure rates.
© 2003 Elsevier Science B.V. All rights reserved.

Keywords: Ad hoc networks; Information dissemination; Low energy consumption

1. Introduction

Flooding a message across a network is used in several protocols because it can provide several useful applications. One application is to broadcast an announcement message to every node in a network. An example is flooding a link state message in the OSPF routing protocol [1]. Another application is to find a node that provides a particular service, has a particular resource, or is with a particular identity. One example is flooding a RREQ node-search message in the AODV routing protocol [2] while another example is advertising a "Service Advertisement" packet across a region of a mobile ad hoc network in the GSD service discovery protocol [3]. Yet another application is to use flooding as a basic message delivery mechanism in a highly-mobile ad hoc network, where a unicast routing path between a source and destination nodes is difficult to create and maintain.

Although the message flooding method is very useful, it may waste many mobile nodes' scarce battery energy. For example, suppose that the target node is near the finding node. Although the target node can be found in only a few hops from the finding node, the target-discovery message is still flooded to every node in the network, causing many nodes' battery energy to be wasted. As such, reducing the number of messages that need to be

E-mail address: shieyuan@csie.nctu.edu.tw (S.Y. Wang).

flooded in a large mobile ad hoc network is important.

This paper proposes an approach in which several schemes are used to reduce flood messages' energy consumption without increasing their delivery failure rates. These schemes are called the "merge", "doubleSend", and "filter" scheme respectively in this paper. The merge scheme saves energy by merging several small flood messages into a larger one. This can amortize the fixed and high energy overhead associated with every wireless transmission [4] over several flood messages. The filter scheme saves energy by limiting the scope of a flood message. This can reduce the number of messages that need to be flooded and thus reduce their energy consumption. The doubleSend scheme is used to detect message loss and perform necessary retransmissions. This can maintain flood messages' delivery failure rates to a low value while the merge and filter schemes are used.

This paper compares the performances of these schemes to those of the original approach in various conditions. The rest of this paper is organized as follows. Section 2 surveys related work. Section 3 presents the design and implementation of the proposed schemes. Section 4 compares the performances of the proposed schemes to those of the original approach. Section 5 discusses some issues about the proposed scheme. Section 6 discusses future work. Finally, Section 7 concludes the paper.

2. Related work

To reduce the number of flood messages in a network, the expanding ring search (ERS) method (also called "TTL scoping") has been proposed in the literature. In this method, the finding node sets the Time-to-Live (TTL) value of the target-discovery message to an initial value. If no reply is received within the discovery period, the next target-discovery message is flooded with a TTL value increased by an increment value. This process of increasing the TTL value continues until the target node is found.

The ERS method can reduce the number of flood messages and thus save energy in cases when the target node is close to the finding node. However, the ERS method may use more flood messages and thus more energy consumption than the plain flood method in cases when the target node is far away from the finding node.

In [5], the authors proposed query localization techniques for on-demand routing protocols. To set up a unicast routing path to a target node, the source node first floods a query message across the network. When the reply message is sent back to the source node, every node on the routing path needs to learn from this reply message and remember that it is on the routing path between the source and target nodes. Later on, when the path breaks, the source node will re-flood the query message, inside which a counter is used to limit the scope of the flood messages.

When a node receives a flood message, it first checks if it is on the routing path of this particular (source, target) flow. If yes, it continues flooding the message. Otherwise, it decrements the value of the counter by one and then floods it. When a node receives a flood message whose counter value already drops to zero, it discards the flood message without flooding it. By adjusting the initial counter value, the method can control the scope of the flood message. Using a smaller value will result in a smaller scope because the flood messages cannot go far away from the previous routing path. However, doing so will increase the message delivery failure rate.

Since this approach requires a node to remember whether it is on the routing path of a particular (source, target) flow, this approach needs to recognize the reply message format of the routing protocol which it works with. As such, it is hard to be used as a generic traffic-reducing method.

In [6], the authors proposed several approaches to mitigate the broadcast storm problem. These approaches try to reduce the number of flood messages in a network. Instead of having each mobile node continue to broadcast a received flood message, some heuristics are used to prohibit some mobile nodes from doing it. These approaches can reduce the number of flood messages at the cost of a higher message delivery failure rate. Because these approaches are aimed to reducing the number of flood messages that are spawned from the same flood message issued by a mobile node, these approaches can be viewed as intra-broadcast storm traffic-reducing methods.

In contrast, the approach proposed in this paper is generic and can be viewed as an inter-broadcast storm traffic-reducing method. In this approach, when there are many broadcast storms going on at the same time, they are combined (merged) together to form a single broadcast storm to reduce the number of flood messages in the network. As such, the energy consumption of mobile nodes and the degree of packet contentions and collisions at the MAC layers can be greatly reduced. This will result in a smaller flood message delivery failure rate.

In [7], the authors proposed the IMEP protocol for control packet aggregation and encapsulation. Although in this paper the authors defined the IMEP message formats and usages, they did not provide any performance evaluation of the IMEP protocol. In contrast, in this paper we not only define the message format and architecture of the merge scheme, we also evaluate its performance gains in various conditions.

3. The proposed approach

In our approach, three schemes are used on each mobile node. We call these schemes the "merge", "doubleSend", and "filter" scheme, respectively. An architecture that uses all of these schemes is depicted in Fig. 1. The details of each scheme are explained below.

3.1. The merge scheme

The merge scheme tries to merge multiple small flood messages into a larger one. This can reduce energy consumption because the fixed and high energy overhead associated with every wireless transmission can be amortized over many flood messages.

3.1.1. Architecture

In a normal network where the merge scheme is not used, we envision that, for each target or resource discovery service, there will be a corresponding flood server running on every mobile node. This service-specific flood server (abbreviated as SSFS for simplicity) will need to understand the packet format defined for this particular service and use a method to determine whether to continue flooding the message across the network. For example, to support the AODV target node discovery service, every mobile node needs to run an AODV flood server that understands the AODV's RREQ packets and knows whether to continue flooding the message.

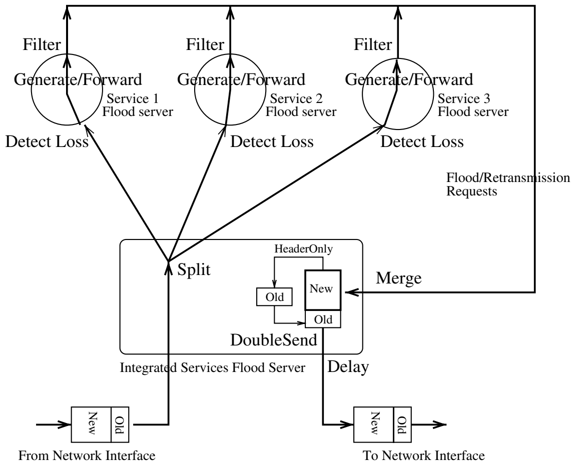

Fig. 1. The architecture of the proposed approach. Three schemes (merge, doubleSend, and filter) are used in this architecture.

When the merge scheme is used in a network, an "integrated-service flood server" (abbreviated as ISFS for simplicity) will be run on every mobile node. Each of the SSFSs that are already running on a mobile node needs to register with the ISFS and make a TCP connection to the ISFS. The TCP connection between a SSFS and the ISFS is used by them to exchange flood messages. We choose to use the TCP protocol because it provides reliable transfers. If instead the light-weight UDP protocol is used, when the ISFS is temporarily overloaded, messages sent from ISFSs may be dropped because the ISFS has no time to process them.

For the output direction, a SSFS now only needs to (1) determine whether a received flood message needs to be further flooded and (2) generate flood messages that should be flooded further (these messages include the messages generated by the SSFS itself and the messages that need to be forwarded further by this SSFS). The job of transmitting generated flood messages through the wireless interface is delegated to the ISFS. When the ISFS receives a flood message from a TCP connection between it and a SSFS, the ISFS will purposely delay the flood message in a merge queue. After collecting enough flood messages or a certain period of time has elapsed, the ISFS will merge the flood messages in the queue to form a larger "integrated-service flood message" (abbreviated as ISFM for simplicity). The ISFS will then flood the ISFM through the wireless interface by broadcasting it as a single message.

For the input direction, when the ISFS receives an ISFM, it will send a copy of the ISFM to each involved SSFS (through the TCP connection between it and the involved SSFS). For security reasons, the copy sent to a SSFS will contain only the flood messages that belong to the SSFS. When a SSFS receives a copy of an ISFM, which contains only the flood messages that belong to this SSFS, the SSFS will process each of these flood messages one by one as if the merge scheme was never used. For each flood message, the SSFS will first determine whether the message needs to be further flooded. If yes, it generates a flood message (or just modifies the current flood message) and then sends it to the ISFS.

The architecture used by the merge scheme is depicted in Fig. 1. This architecture not only can be used to merge flood messages that belong to different services, it can also be used to merge flood messages that belong to the same service but belong to different broadcast storms initiated by different mobile nodes.

3.1.2. Message format

In the merge scheme, a flood message is composed of a "flood message common header" (FMCH) and a "flood message body" (FMB). The FMB is the body of a flood message and contains service-specific information. A flood message's FMB may be empty if its FMCH already contains enough information. The FMCH, depicted in Fig. 2(b), contains several general-purpose fields. It is aimed to supporting all kinds of SSFS. In the following, we will explain the usage of each field.

The Service field indicates the service that this flood message belongs to. The SrcIP field stores the IP address of the source node that initiates this flood message. The Seq# field stores the sequence number assigned by the source node to this flood message. The DstIP field stores the IP address of

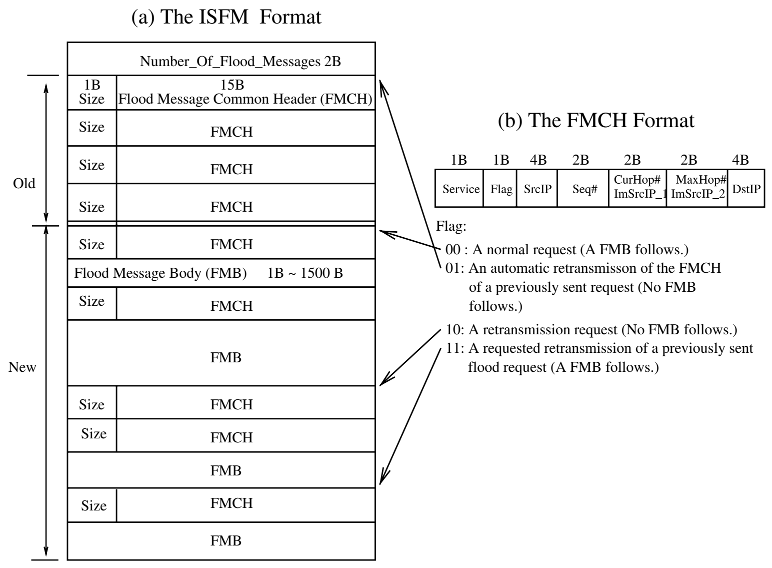
(a) The ISFM Format
Number_Of_Flood_Messages 2B
1B Size
15B Flood Message Common Header (FMCH)
Size
FMCH
Flood Message Body (FMB) 1B ~ 1500 B
FMB
Old
New
(b) The FMCH Format
1B 1B 4B 2B 2B 2B 4B
Service
Flag
SrcIP
Seq#
CurHop# ImSrcIP_1
MaxHop# ImSrcIP_2
DstIP
Flag:
00 : A normal request (A FMB follows.)
01: An automatic retransmisson of the FMCH of a previously sent request (No FMB follows.)
10: A retransmission request (No FMB follows.)
11: A requested retransmission of a previously sent flood request (A FMB follows.)

Fig. 2. The formats used by "service-specific flood messages" and "integrated-service flood messages".

the desired target node. (Note: if the flood message is an announcement destined for all nodes in a network, the DstIP field is set to all zeros to indicate this intention.) The CurHop# field stores the distance in hops from the source node to the node that receives this flood message. This field is initially set to 1 by the source node before transmitting the flood message to the network. In the network, when a SSFS receives a flood message, it increases this field by one before continuing to flood this message. The MaxHop# field functions like the TTL field in the IP header. The source node sets this field to limit the maximum scope a flood message can expand (like the ERS scheme). The Flag, ImSrc_IP1, and ImSrc_IP2 fields are used by the doubleSend scheme. We will explain their usages later.

The format of an ISFM is depicted in Fig. 2(a). The Number_of_Flood_Messages field stores the number of flood messages that are included in this ISFM. An ISFM is divided into two sections. The "new" section stores complete flood messages. (A flood message is said to be complete here if its FMCH and non-empty FMB are present at the same time.) The "old" section stores only the FMCHs of the flood messages that were previously sent. It is used by the doubleSend scheme to detect flood message losses. The Size field, which is next to a flood message's FMCH, specifies the total size of that flood message. With this field, the size of a flood message's FMB can be arbitrary.

When it is time to send out an ISFM, the ISFS will pack as many flood messages as possible into the ISFM, subject only to the MTU constraint of the wireless interface. In our study, we used 1500 bytes as the MTU, because it is the MTU used by most IEEE 802.11(b) NICs (e.g., Lucent's Orinoco and Cisco's Aironet) on FreeBSD and Linux platforms. How often to send out an ISFM is a system parameter and can be varied. For example, we can use a fixed-delay scheme in which the merge delay is always set to 20 ms. Or we can use a controlled-load scheme to limit the bandwidth consumed by flood messages to a threshold (e.g., 20% of the link bandwidth). That is, as long as the bandwidth consumed by flood messages is less than the specified threshold, the merge delay can be dynamically shortened to a small value to reduce the merge degree. This will result in small merge delay under light traffic.

We note that the above merge delay parameter represents only the maximum amount of time between sending two successive ISFMs. Whenever the total size of the flood messages waiting in the merge queue exceeds the MTU (1500), the ISFS will form an ISFM to send out some flood messages. The sending rate is limited only by the underlying MAC layer due to bandwidth usage, packet collisions, or signal interferences. Because there is a maximum queue length limitation on the merge queue (it is set to 50 flood messages in our study), when the merge queue is full, flood messages sent from SSFSs will be dropped by the ISFS.

3.1.3. Advantages

When the FMBs of flood messages are small (for example, the AODV's RREQ is 24-byte long), the merge scheme provides two advantages. First, it reduces the battery energy consumed by flood traffic. Second, it reduces the degree of packet contentions and collisions at the MAC layer, which will decrease the flood message delivery failure rate.

The battery energy saving is achieved because sending multiple small messages as a single one can amortize the fixed and high energy overhead associated with every wireless transmission. According to [4], the energy ($\text{mW} \times \text{s}$) consumed by transmitting or receiving an IEEE 802.11b [8] packet can be modeled as a linear equation: $\text{Energy} = m \times \text{size} + b$, where b is a fixed component associated with device state changes and m is an incremental component which is proportional to the size (in bytes) of the packet. The (m, b) for transmitting and receiving a broadcast IEEE 802.11b packet are modeled in [4] as (2.1, 272) and (0.26, 50), respectively.

The degree of packet contentions and collisions at the MAC layer is reduced because the number of flood messages that need to be transmitted at the MAC layer is reduced. It is a natural result.

3.1.4. Disadvantages

Although the merge scheme has several advantages, it has two disadvantages. The first disadvantage is that the per-hop-delay experienced by flood messages is larger. The second disadvantage is that multiple flood messages may be dropped at the same time.

The increased per-hop-delay is a cost that the merge scheme must pay for achieving the advantages discussed above. To increase the merge degree, we would like to set the merge delay to a larger value. However, this would increase the per-hop-delay experienced by flood messages.

Although the delay performance of the merge scheme is worse than that of the original scheme (i.e., a network that does not use the merge scheme), our simulation results show that, if the load of flood traffic keeps increasing, packets will start to be queued in the wireless interfaces' FIFOs. At this time, the per-hop-delay of the original scheme will also become large and not far from the per-hop-delay of the merge scheme.

In the merge scheme, multiple small flood messages may be dropped at the same time. The reason is that multiple small flood messages are merged and transmitted together as a single flood message. If the message experiences packet collision or corruption, all of these small flood messages will be lost as well. Also, a large message is more likely to get corrupted than a small message under a non-zero bit error rate (BER) condition. These factors may result in a higher flood message delivery failure rate than the original scheme.

In our simulation results, however, the flood message delivery failure rate of the merge scheme is smaller than that of the original scheme. This is due to a large reduction of packet contentions and collisions at the MAC layer. This benefit offsets the bad effect caused by the "multiple-losses-at-the-same-time" problem.

3.2. The doubleSend scheme

In Section 3.1.4, we discussed the "multiple-losses-at-the-same-time" problem in the merge scheme. To mitigate this problem, our approach adds a flood message loss detection and retransmission scheme to the merge scheme. We call this scheme the doubleSend scheme.

In the doubleSend scheme, a SSFS will send a flood message twice in its two consecutive ISFMs. In the first ISFM, both the FMCH and FMB of the flood message are transmitted in the new section of the ISFM. The Flag field of the FMCH is set to 00 to indicate that this is a new transmission. In the second ISFM, however, only the FMCH of the flood message is transmitted in the old section, and the Flag field is set to 01 to indicate that this is an old transmission. The new section is for normal uses and the old section is for flood message loss detection. Fig. 2(a) shows the arrangement.

When a SSFS receives an ISFM, it processes the complete flood messages in the new section in the normal way. In addition, it stores these processed flood messages into its cache, which is used to store recently-received flood messages. The SSFS uses the FMCHs in the old section to check whether it had successfully received these flood messages. For each FMCH in the old section, if the SSFS cannot find the FMCH in its cache, the flood message with this FMCH apparently was lost.

When detecting a flood message loss, the SSFS constructs a retransmission request asking one (or multiple) SSFS(s) to retransmit the flood message with the specified FMCH. The retransmission request is simply the FMCH of the lost flood message. The only exception is that its Flag field now is set to 10 to indicate that it is a retransmission request. To specify which SSFS(s) should perform the retransmission, the IP address of the desired SSFS is entered into the ImSrc_IP1, and ImSrc_IP2 fields. This retransmission request then is sent to the ISFS, where it will be merged with other flood messages and put into the new section of the next outgoing ISFM.

Normally, the retransmission of a lost flood message is performed by only one SSFS to save wireless bandwidth and battery energy. (Normally it is the SSFS whose message makes this SSFS detect this flood message loss.) However, in a fast-moving mobile ad hoc network, the SSFS that detects a flood message loss may want to ask all of its neighboring SSFSs to perform the retransmission to increase the retransmission success rate. In this case, the ImSrc_IP1 and ImSrc_IP2 fields can be set to all ones to indicate this intention. In our simulation study, in the cases when the moving speed of mobile nodes is set to 1 m/s, we let the retransmission of a lost flood message be performed by one SSFS. In the cases when the moving speed is set to 40 m/s, we let the retransmission be performed by neighboring SSFSs.

When a SSFS receives a flood message retransmission request (in the new section of the just-received ISFM), it first checks whether it is the node who should be responsible for the retransmission. If yes, then it checks its cache to see whether the requested flood message is still kept in the cache. If yes, then it retrieves the requested flood message from the cache, sets its Flag to 11 to indicate that it is a retransmission, and then sends it to the ISFS. The ISFS will then merge it with other flood messages and put it into the new section of the next outgoing ISFM.

The traffic of retransmission requests and retransmissions can be reduced by several methods implemented in the ISFS. For example, one method that can eliminate unnecessary retransmission requests (and thus the retransmissions caused by them) is described below. When the ISFS receives an ISFM, for each complete flood message in the new section, it extracts the (Service, SrcIP, Seq#) information and compares the information to each retransmission requests waiting in the merge queue. If there is a match, the retransmission request is canceled because the ISFS will soon deliver the requested retransmission to the requesting SSFS.

The cost of using the doubleSend scheme is minimal. To enable a SSFS to detect flood message losses, the FMCH of each flood message is sent twice. Since the size of the FMCH is small (only 16 bytes in this approach) and it is transmitted with other flood messages as a single ISFM, the extra bandwidth and energy consumed by these FMCHs are little.

We note that in the current design, if two successive ISFMs are lost, the loss of a flood message carried in the first ISFM will not be detected and thus be retransmitted. In addition, when a SSFS receives a retransmission request but finds that the requested flood message is no longer in its cache, the request will simply be discarded.

3.3. The filter scheme

The filter scheme is composed of the band and continueSend techniques, whose details are explained in Sections 3.3.1 and 3.3.2, respectively. The filter scheme can reduce the scope of a flood message in the network without increasing its delivery failure rate.

In Section 3.1.3, we show that the merge scheme can significantly reduce energy consumption when the average size of FMBs is small. However, as the average size of FMBs increases, the savings will become smaller and finally become insignificant. To effectively reduce energy consumption when the average size of FMBs is not small, the filter scheme is proposed to reduce the amount of traffic that needs to be flooded in a network.

To support the filter scheme, every node in the network needs to keep a control block for every active node in the network. The usages of the fields of a control block are explained below. For a control block, we call the node that creates this control block the "local" node, and the node that this control block is created for the "remote" node.

IPaddr: The IP address of the remote node.
Seq#: The highest sequence number from the remote node that has been seen by the local node.
DistInc: The distance in hops from the local node to the remote node.
DistDec: The distance in hops from the local node to the remote node.

Although both the DistInc and DistDec fields store the distance in hops from the local node to the remote node, their usages and functions are different. In the following, we describe how a local node updates the information kept in a control block.

Initially, for every created control block, the DistInc is set to 255 and DistDec is set to 0. (The number 255 represents the maximum distance in hops used in our approach.) When a flood message arrives, the local node uses the SrcIP field in the flood message's FMCH to locate the corresponding control block. Once found, the local node compares the Seq# value in the flood message's FMCH to the Seq# value in the control block to determine whether this newly-arrived flood message is a fresher message from the remote node.

If the newly-arrived Seq# value is greater than the one kept in the control block, since this flood message is a fresher flood message from the remote node, both the DistInc and DistDec values are set to the CurHop# value in the FMCH to reflect the most recent distance between the remote and local node. On the other hand, if the newly-arrived Seq# value is equal to the kept one, the DistInc value in the control block is further compared to the CurHop# value in the FMCH. If the DistInc value is greater than the CurHop# value, both the DistInc and DistDec values are set to the CurHop# value in the FMCH. (Note: if the newly-arrived Seq# value is less than the kept one, the newly-arrived flood message is discarded and not processed.)

The CurHop# in the FMCH carries the distance in hops between the remote node and the local node. Because a flood message may result in many spawned messages and each of these messages (with the same Seq#) may take a different path from the remote node to this local node, the above processing will obtain the most recent minimum distance between the remote and the local node.

Periodically, a timer will expire and its routine is executed to age out the DistInc and DistDec values. In every execution of the timer routine, the DistInc value is increased by one and the DistDec value is decreased by one. If no fresher flood message arrives to update the DistInc and DistDec values, the DistInc value will finally go up to 255 and the DistDec value will finally go down to 0. The age-out period is a system parameter and can be varied. For a fast-moving mobile ad hoc network, we may want to set the age-out period to a small value so that old distance information can be aged out soon. On the contrary, a large age-out period can be used for a slowly-moving mobile ad hoc network. In our simulation study, we set the period to 3 s.

The DistInc and DistDec values are aged out in opposite directions. This is because they are used for different purposes. Their usages will become clear in the next section.

3.3.1. The band technique

The band technique uses three filtering methods to limit the scope of a flood message to only a narrow band between the source and target nodes.

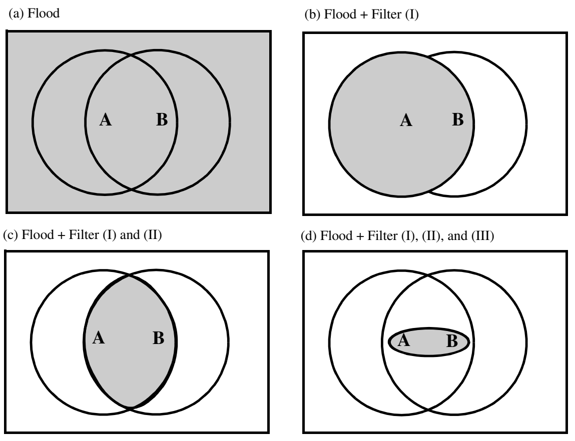
(a) Flood
(b) Flood + Filter (I)
(c) Flood + Filter (I) and (II)
(d) Flood + Filter (I), (II), and (III)
A
B

Fig. 3. The scope of a flood message when the three filtering methods are used.

This can greatly reduce the amount of flood traffic in the network. In the following, we use the notation A.DistDec(B) to represent the DistDec value in the control block that is created for node B and created on node A. A.DistInc(B) is used in a similar way.

Suppose that node A issues a flood message that is destined for node B, and C is a third-party node in the network. In the following, we present the three filtering methods and compare their performances to that of the plain flood method.

In the plain flood method, the flood message will be disseminated to all nodes in the network. Fig. 3(a) shows the scope of this dissemination.

The first filtering method is to set the MaxHop# in the FMCH of the flood message to A.DistInc(B). Because every node, such as node C, will drop a received flood message whose current CurHop# value is greater than its MaxHop# value, the flood scope will be limited to a circle centered at node A. (This same effect can be achieved by the ERS method.) Fig. 3(b) shows the scope of this dissemination.

The second filtering method is that, when node C receives a flood message, if C.DistDec(B) is greater than the MaxHop# value specified in the flood message's FMCH, node C drops this flood message. The rationale behind this method is as follows. If node A thinks that node B is within K hops from it and thus sets the MaxHop# to K, a node (say C) that is more than K hops away from node B definitely is not on the right path from node A to node B. As such, node C should discard the received flood message and need not further flood it. This method further limits the scope to the intersection of the two circles. Fig. 3(c) shows the scope of this dissemination.

The third filtering method is that, when node C receives a flood message, if ((C.DistDec(B) + C.DistDec(A)) > (MaxHop# + ExtraHops)), where ExtraHops is a system parameter and is set to 2 in our simulation study, node C should drop this flood message. The rationale behind this method is as follows. To reduce the number of spawned flood messages, we should let only the nodes on the band between node A and node B perform the flood operations. For node C, if the sum of the distance between C and A and the distance between C and B exceeds the MaxHop# value too much, node C is unlikely on the band. As such, node C should discard the flood message. The ExtraHops system parameter is used to control the width of the band. Using a smaller value for it can eliminate more flood traffic at the expense of a higher flood message delivery failure rate. This method further limits the scope to just a narrow band between node A and node B. Fig. 3(d) shows the scope of this dissemination. We see that using these three filtering methods can significantly reduce the amount of traffic that needs to be flooded in a network.

After explaining these filtering methods, now it is clear why the DistDec and DistInc values of a control block are updated in opposite directions. The reason is that although we want to use these methods to reduce the amount of flood traffic in a network, we do not want to increase the flood message delivery failure rate too much at the same time. Gradually decreasing the DistDec value and increasing the DistInc value when no fresher flood message arrives is a conservative measure. This can ensure that the flood message delivery failure rate will not be increased too much.

For the filter scheme to perform perfectly, a node will need to have the most up-to-date information about the distance between it and a target node. As such, an active node will need to periodically perform a full-scope announcement so that all other nodes can learn the distance between them and this node. Certainly this period depends on the moving speed of mobile nodes. A slow-moving mobile ad hoc network can use a longer period to reduce the full-scope flood overhead. The overhead can also be reduced by periodically piggybacking the announcement on a to-be-sent normal flood message. Since the announcement is very small (i.e., a 16-byte FMCH), when it is merged with other normal flood messages and transmitted together in the network, the additional energy overhead caused by flooding this announcement is tiny.

3.3.2. The continueSend technique

The continueSend technique works with the first filtering method presented in Section 3.3.1 to reduce a flood message's delivery failure rate.

In the first filtering method, a flood message is dropped when its current CurHop# value is greater than the value specified in the MaxHop# field. Although this method effectively reduces the amount of traffic caused by flooding a message, it can easily increase the flood message's delivery failure rate. This problem may likely happen in a fast-moving mobile ad hoc network. The reason is that, in such a network, the information about the distance between two nodes may quickly become outdated.

To solve this problem, the continueSend technique is proposed. When a node (say node C) determines to drop a flood message that is initiated by node A and destined for node B, it does one more check. If (C.DistInc(B) < Neighborhood), where Neighborhood is a system parameter and is set to 3 in our simulation study, this flood message is not dropped. Instead, node C increases the MaxHop# value of this flood message by one and continues to flood it. Doing this will give the flood message one more chance to get closer to its final destinaton node. The rationale behind this technique is as follows. If a flood message is already very close to its target node, we should help it to reach its target node, since the bandwidth cost for doing this job is minimal.

Fig. 4 graphically presents the continueSend technique. In the figure, because the target node (labeled as "Dst") moves very fast, which makes the distance information kept on the source node

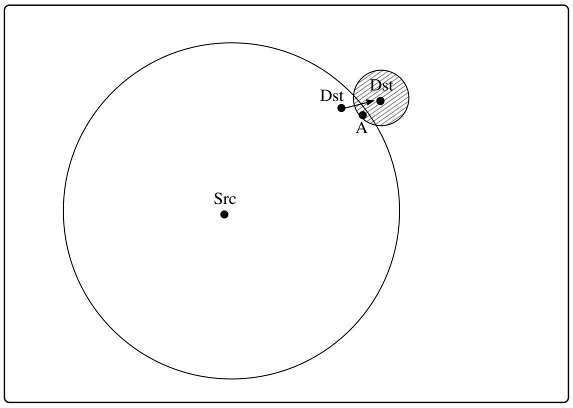

Fig. 4. In the continueSend technique, node A will give a should-be-dropped flood message one more chance to get closer to its target node.

obsolete, the flood message issued by the source node cannot reach it. When the continueSend technique is used, node A will continue flooding this flood message and give it one more chance to reach its target node. We can see that, because the scope of the target node's neighborhood is set to a small value, using the continueSend technique does not spawn many unnecessary flood messages.

4. Simulation results

The proposed approach uses several schemes to reduce the energy consumption of mobile nodes. To understand the relative benefits offered by each of these schemes, we built four different approaches and compared their performances. The first approach, named "original", is the original (plain) flood approach. The second approach, named "merge", uses the merge scheme on top of the original approach. The third approach, named "merge + doubleSend", uses the doubleSend scheme on top of the merge approach. The fourth approach, named "merge + doubleSend + filter", uses the filter scheme on top of the merge + doubleSend scheme.

4.1. Simulator settings

We implemented the proposed approach in the NCTUns 1.0 network simulator [9], which is an extensible and high-fidelity network simulator just released to the networking community on 11/01/2002. The program code of the NCTUns 1.0 network simulator can be downloaded at http://NSL.csie.nctu.edu.tw/nctuns.html.

In simulations, the transmission and interference ranges of wireless interfaces are set to 250 and 550 m, respectively. In the study, each simulation runs 300 simulated seconds and each data point represents the average of five runs. The details of simulation settings are presented below.

4.1.1. Network configuration

The network is composed of 128 fixed wireless relay nodes and 384 mobile nodes. As such, 3/4 nodes in the network are mobile nodes. The relay nodes are placed on the grids of a 16×16 lattice using a particular pattern shown in Fig. 5. The size of the network field is 3940 (m) $\times$ 3940 (m). The distance between two diagonally neighboring relay nodes is set to 240 m so that they can communicate with each other directly.

This network configuration is particularly chosen such that no matter how and where these mobile nodes move to, the network is connected (not partitioned) all the time. This property is

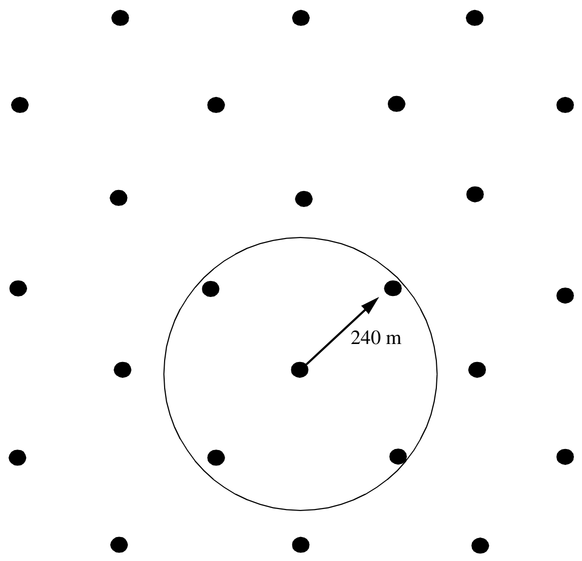

Fig. 5. Fixed wireless relay nodes are placed on the grids of a 16×16 lattice using this pattern. Here only a part (7×7) of the lattice is shown.

important for us to compare the flood message delivery failure rate performance of each tested approach. Without this property, some flood message delivery failures may be caused by network partitions, not by the design of these approaches. As such, the flood message delivery failure rates reported by the simulator will not be able to accurately reflect the capability of these tested approaches. We note that the effects of these fixed nodes on the failure rates of all approaches are the same. This is because the mobile nodes' moving scenarios and their relative positions to these fixed nodes over time are exactly the same in all approaches.

4.1.2. Moving pattern

For mobile nodes, we use a random way-point moving pattern for them. The moving pattern of a node is a series of turning points, whose locations are randomly chosen. With a specified moving speed, the mobile node moves from the current turning point to the next turning point without any pause. At the beginning of each simulation, these mobile nodes are randomly placed in the network field.

4.1.3. Traffic pattern

Roughly at the end of every second, each node (either a fixed relay node or a mobile node) uses a probability to determine whether it should issue (i.e., generate) a flood message at this time. This probability is not the probability for a node to forward a flood message. The target node of the flood message is set to a mobile node and is randomly chosen from all mobile nodes. To avoid synchronization of flood traffic, the clocks used by the nodes in a network are offset by random numbers.

4.1.4. System settings

Every wireless interface has a FIFO queue to hold its outgoing packets that cannot be sent out at this moment. The maximum queue length for these FIFOs are set to a small value of five packets to avoid a large queuing delay that packets may experience in a FIFO. All merge-related approaches use a fixed merge delay of 20 ms.

4.1.5. System variables

The system variables used in the simulation study include the following variables. The first is the probability for a node to issue a flood message at the end of every second. This variable is used to vary the flood traffic load imposed on the tested network and is varied from 10%, 20%, ..., to 100%. The second variable is the average size of a FMB. This variable has two different values of 16 bytes and 128 bytes. It is used to test the performance of these approaches under different FMB sizes. The third variable is the moving speed of mobile nodes. It has two different values of 1 m/s and 40 m/s. (We also has the 20 m/s results. However, due to space limitation, there is no space to present them.) This variable is used to test the performances of these approaches in slowly-moving and fast-moving mobile ad hoc networks.

4.1.6. Performance metrics

To evaluate the performances of these approaches, we used the following four performance metrics:

1. The first metric is the average energy consumption of all nodes in a network that is consumed by flood traffic. The used energy consumption model is taken from [4].
2. The second is the average flood message delivery failure rates of all nodes in the network (plotted in the log scale).
3. The third is the average per-hop-delay experienced by all flood messages in the network. This value is calculated as follows. First, we sum the finding times of all successful flood messages into S. (A flood message is considered successful if it can reach its target node. In this case, the finding time of the flood message is the elapsed time between when it is issued and when it reaches its target node.) Then we sum the hop count experienced by each successful flood message into H. The per-hop-delay then is calculated as S/H.
4. The fourth is the average packet collision rate of all nodes in a network. This value is calculated as follows. First, we sum the number of packet collisions of all nodes in a simulation run into C. Then we divide C by the product

of the number of simulated seconds of the run and the number of nodes.

4.2. Speed = 1 m/s, FMB = 16 bytes

In this test suite, the moving speed of mobile nodes is set to 1 m/s and the size of a FMB is set to 16 bytes.

Fig. 6 shows the average energy consumption (consumed by flood traffic) of each approach. We see that all merge-related approaches outperform the original approach and have similar performances. If we compare the energy consumption of the original and the merge + doubleSend + Filter approach, the maximum saving $100(1 - 73/245) = 71\%$ is achieved when the probability is 100%.

Fig. 7 shows the average flood message delivery failure rate of each approach. We see that the failure rate of the merge approach is smaller than that of the original approach. This shows that, in a slowly-moving mobile ad hoc network, the advantages caused by a reduction of packet contentions and collisions at the MAC layer is greater than the disadvantage caused by the "multiple-losses-at-the-same-time" problem.

The performance of the merge + doubleSend approach is much better than that of the merge approach. This shows the effectiveness of the doubleSend scheme. The performance of the merge + doubleSend + filter approach is slightly worse than that of the merge + doubleSend approach. This result is expected as the filter scheme may sometimes make mistakes when aggressively eliminating flood traffic. Comparing the failure rates of the original and the merge + doubleSend + Filter approach, we see that it is reduced by a factor of 76 when the probability is 100%.

Fig. 8 shows the average packet collision rate of all nodes of each approach. We see that the collision rate of the original approach is much higher than those of all merge-related approaches. For example, the collision rate of the original approach is 8.75 times higher than those of all merge-related approaches when the probability is 100%. This fact

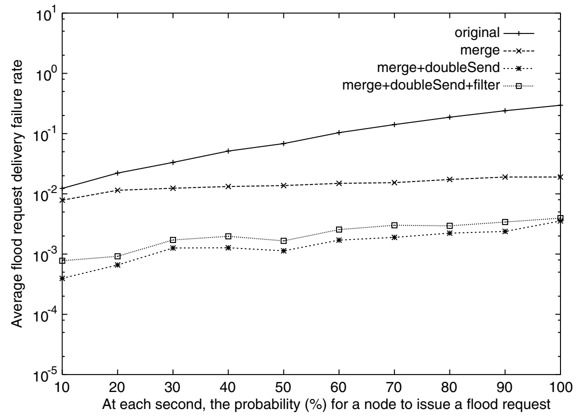

Fig. 7. The average flood message delivery failure rate. Speed = 1 m/s, FMB = 16 bytes.

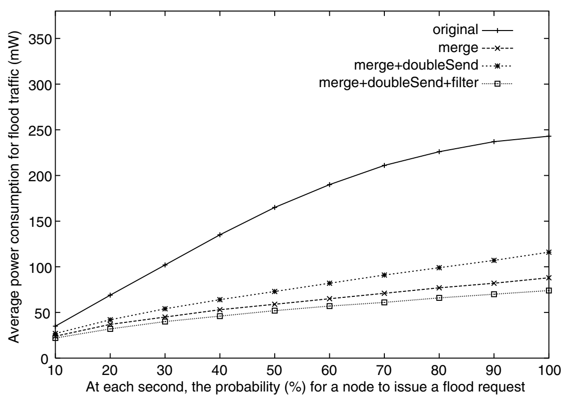

Fig. 6. The average energy consumption consumed by flood traffic. Speed = 1 m/s, FMB = 16 bytes.

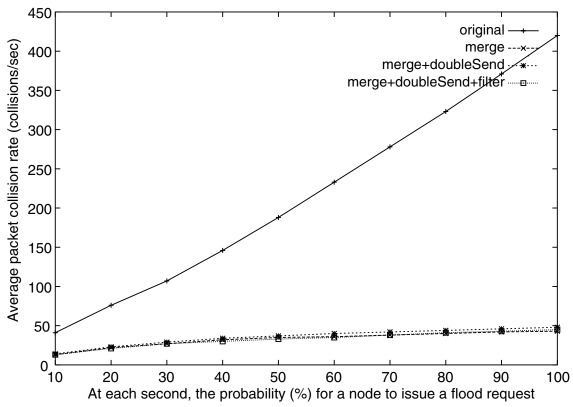

Fig. 8. The average packet collision rate of all nodes. Speed = 1 m/s, FMB = 16 bytes.

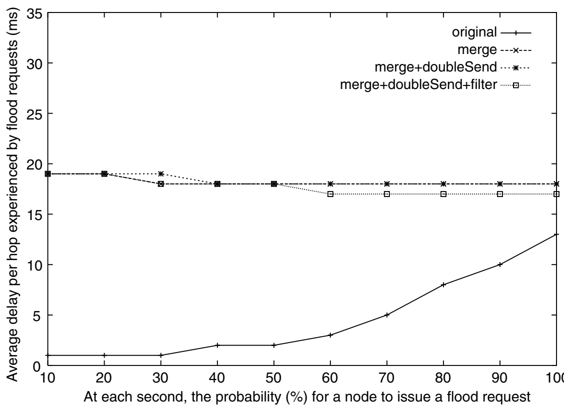

Fig. 9. The average per-hop-delay experienced by flood messages. Speed = 1 m/s, FMB = 16 bytes.

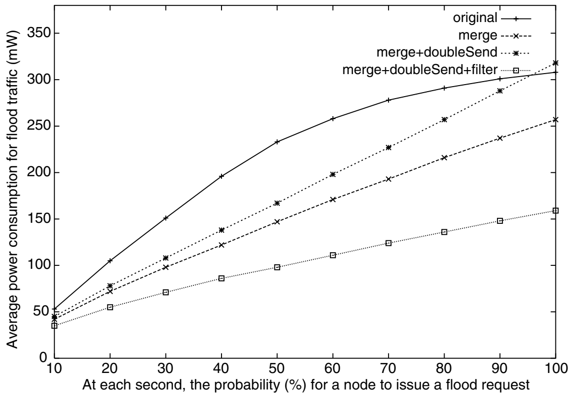

Fig. 10. The average energy consumption consumed by flood traffic. Speed = 1 m/s, FMB = 128 bytes.

shows the effectiveness of the merge operation on reducing the packet contentions and collisions at the MAC layer.

Finally, Fig. 9 shows the average per-hop-delay experienced by flood messages in each approach. We see that the delay is about 20 ms in the merge-related approaches. This result is expected because our approach uses 20 ms as the merge delay. The original approach has a much lower delay when the flood traffic load is light. However, as the load increases and starts to introduce queuing delays to the network, the delay of the original approach increases as well and finally goes up to 12 ms when the probability is 100%.

4.3. Speed = 1 m/s, FMB = 128 bytes

In this test suite, the moving speed of mobile nodes is set to 1 m/s and the size of FMB is set to 128 bytes. Fig. 10 shows the average energy consumption (consumed by flood traffic) of each approach. Comparing this figure to Fig. 6, we see that as the FMB size increases, the energy saving achieved by the merge scheme decreases, the energy saving achieved by the filter scheme increases, and the energy cost for retransmitting lost flood messages increases.

These results are expected. When the FMB size is large, the merge degree will be small. As such, the energy saving achieved by amortizing the fixed energy overhead associated with every wireless transmission over multiple messages diminishes. On the other hand, because now the FMB size is large, the energy saving achieved by eliminating unnecessary flood messages become significant. The maximum energy saving achieved is 49% when the probability is 100%.

Fig. 11 shows the average flood message delivery failure rate of each approach. The observations are similar to those obtained from Fig. 7. The failure rate is reduced by a factor of 34 when the probability is 100%.

Fig. 12 shows the average packet collision rate of all nodes of each approach. Again, we see that the collision rate of the original approach is much

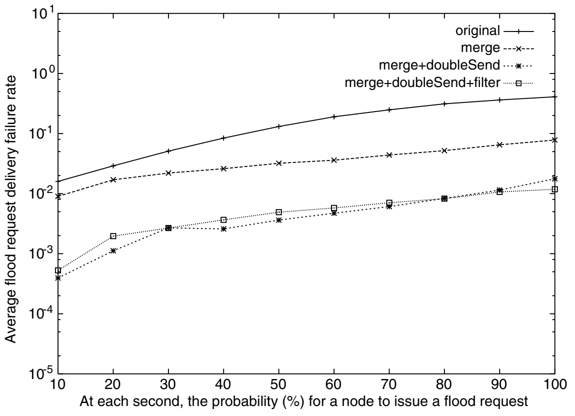

Fig. 11. The average flood message delivery failure rate. Speed = 1 m/s, FMB = 128 bytes.

higher than those of all merge-related approaches, and the maximum collision rate reduction factor is 6.20 when the probability is 100%. Compared this figure with Fig. 8, we see that the collision rates of merge-related approaches slightly increase when the FMB size increases from 16 to 128 bytes. We attribute this phenomenon to the reduction of merge degree, which results in more packets to be sent individually.

Finally, Fig. 13 shows the average per-hop-delay experienced by flood messages in each approach. The observations are similar to those obtained from Fig. 9. However, we see that when the probability is greater than 70%, the average per-hop-delays of the merge and merge + doubleSend approaches start exceeding the 20 ms merge delay. This means that in these load conditions, the network is already overloaded by flood traffic and packets start experiencing queuing delays. We see that because the merge + doubleSend + filter scheme can eliminate more flood traffic, in the same load conditions, the network is still not overloaded. This explains why the per-hop-delay remains about 20 ms in the merge + doubleSend + filter scheme.

4.4. Speed = 40 m/s, FMB = 16 bytes

In this test suite, the moving speed of mobile nodes is set to 40 m/s and the size of FMB is set to 16 bytes. Fig. 14 shows the average energy consumption (consumed by flood traffic) of each approach. The observations about this figure are similar to those obtained from Fig. 6. The maximum energy saving achieved is 67% when the probability is 100%.

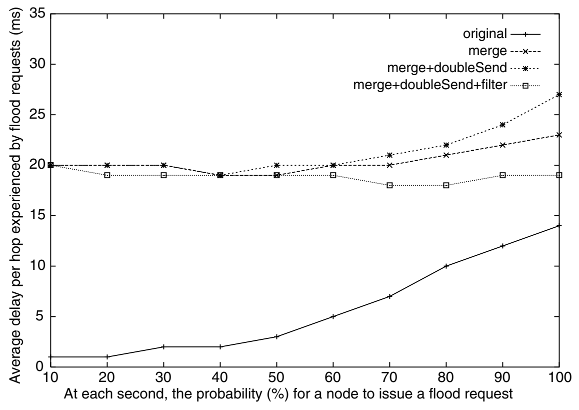

Fig. 13. The average per-hop-delay experienced by flood messages. Speed = 1 m/s, FMB = 128 bytes.

Fig. 15 shows the average flood message delivery failure rate of each approach. The failure rate is reduced by a factor of 40 when the probability is 100%. Comparing this figure to Fig. 7, we see that when the moving speed of mobile nodes increases, the failure rate of the merge + doubleSend + filter approach increases as well. This result is expected. The reason is that, in a fast-moving mobile ad hoc network, the information about the distance between two mobile nodes is likely to become out-

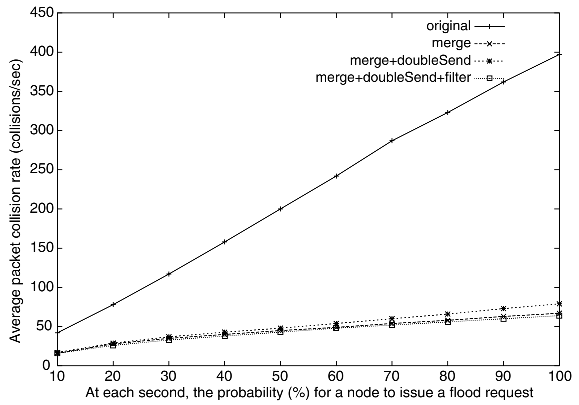

Fig. 12. The average packet collision rate of all nodes. Speed = 1 m/s, FMB = 128 bytes.

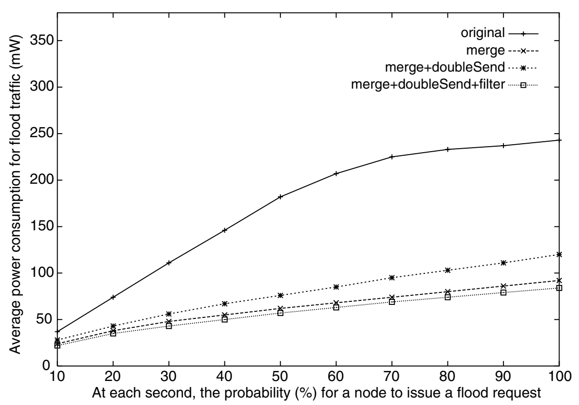

Fig. 14. The average energy consumption consumed by flood traffic. Speed = 40 m/s, FMB = 16 bytes.

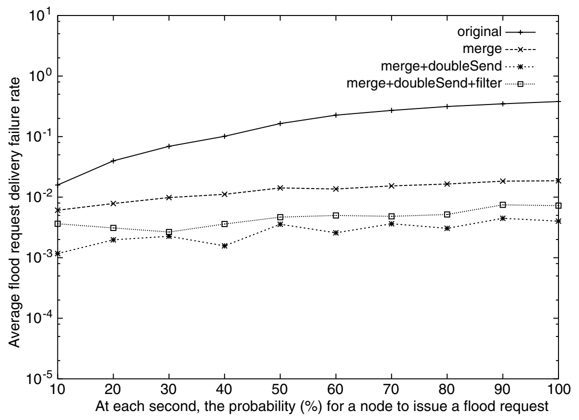

Fig. 15. The average flood message delivery failure rate. Speed = 40 m/s, FMB = 16 bytes.

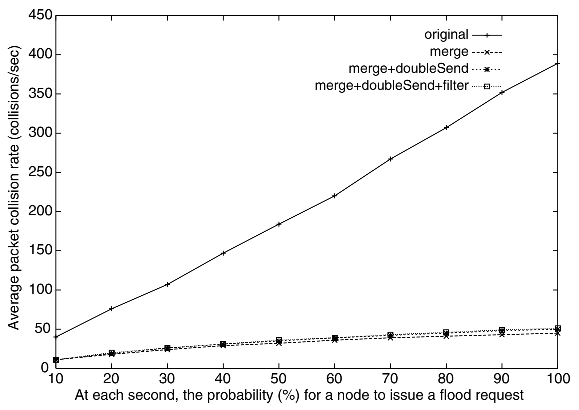

Fig. 16. The average packet collision rate of all nodes. Speed = 40 m/s, FMB = 16 bytes.

dated quickly. This will hurt the effectiveness of the filter scheme. We also see that in light load conditions, the failure rate of the merge approach is higher than the original approach. This shows that, due to an increased packet loss rate, the bad effect of the "multiple-losses-at-the-same-time" problem becomes significant in a fast-moving mobile ad hoc network.

Fig. 16 shows the average packet collision rate of all nodes of each approach. Again, we see that the collision rate of the original approach is much higher than those of all merge-related approaches, and the maximum collision rate reduction factor is 7.62 when the probability is 100%. Compared this figure with Fig. 8, we see that the collision rates of merge-related approaches remain about the same when the mobile nodes' moving speed increases from 1 to 40 m/s. This result is expected as the merge degree in both of these two cases are the same.

Finally, Fig. 17 shows the average per-hop-delay experienced by flood messages in each approach. One interesting phenomenon is that as the flood traffic load increases, the per-hop-delay of these merge-related approaches slightly decreases. To find out the reason, we studied the elapsed time and hop count of every successful flood message. We found that the average hop count of these flood messages is about the same in all levels of traffic load. However, the average finding time of these flood messages slightly decreases as the traffic load increases.

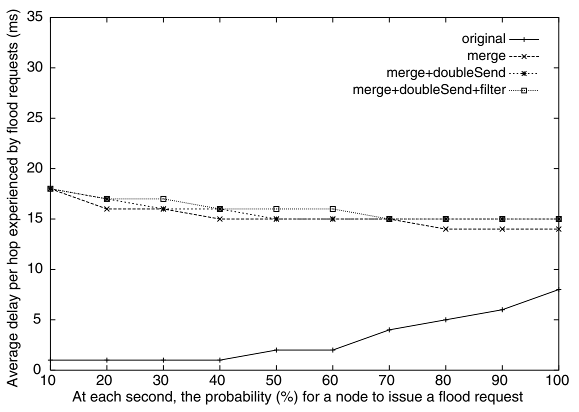

Fig. 17. The average per-hop-delay experienced by flood messages. Speed = 40 m/s, FMB = 16 bytes.

4.5. Speed = 40 m/s, FMB = 128 bytes

In this test suite, the moving speed of mobile nodes is set to 40 m/s and the size of FMB is set to 128 bytes. Fig. 18 shows the average energy consumption (consumed by flood traffic) of each approach. The observations about this figure are similar to those obtained before. The maximum energy saving achieved is 36% when the probability is 100%.

Fig. 19 shows the average flood message delivery failure rate of each approach. The failure rate is reduced by a factor of 22 when the probability is 100%.

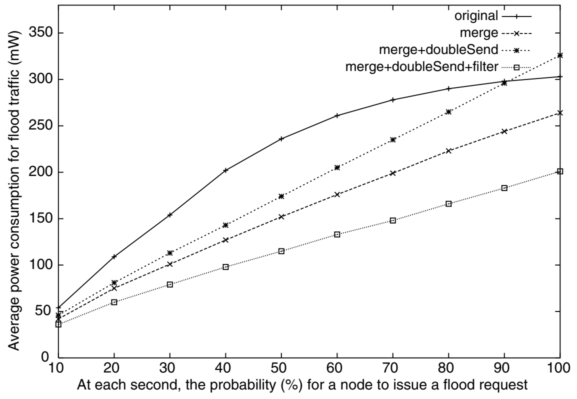

Fig. 18. The average energy consumption consumed by flood traffic. Speed = 40 m/s, FMB = 128 bytes.

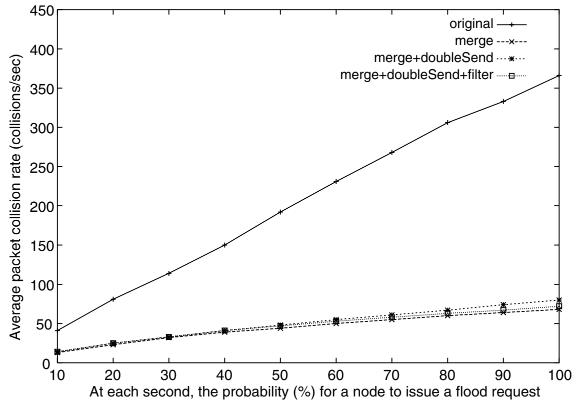

Fig. 20. The average packet collision rate of all nodes. Speed = 40 m/s, FMB = 128 bytes.

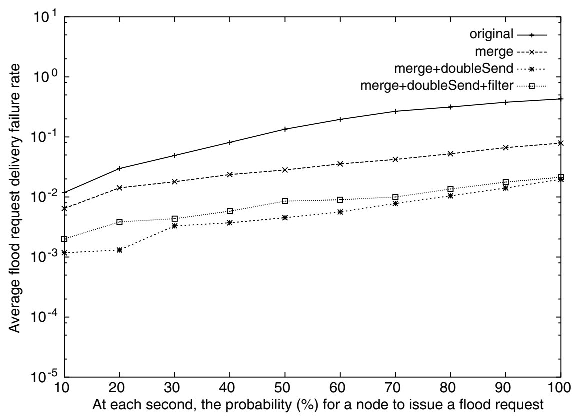

Fig. 19. The average flood message delivery failure rate. Speed = 40 m/s, FMB = 128 bytes.

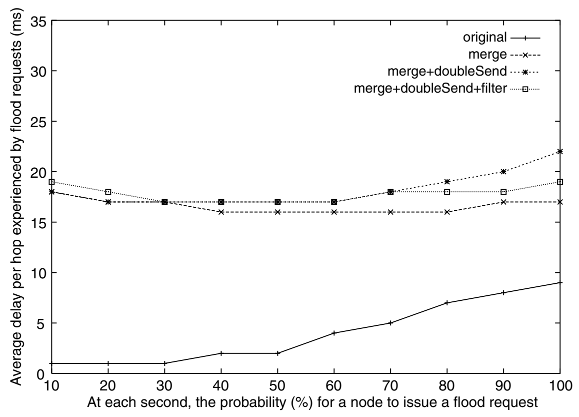

Fig. 21. The average per-hop-delay experienced by flood messages. Speed = 40 m/s, FMB = 128 bytes.

Fig. 20 shows the average packet collision rate of all nodes of each approach. Again, we see that the collision rate of the original approach is much higher than those of all merge-related approaches, and the maximum collision rate reduction factor is 5.38 when the probability is 100%. When comparing this figure with Fig. 16, we see that the collision rates of merge-related approaches slightly increase when the FMB size increases from 16 to 128 bytes, and we attribute this phenomenon to the reduction of merge degree. When comparing this figure with Fig. 12, we see that the collision rates of merge-related approaches remain about the same when the mobile nodes' moving speed increases from 1 to 40 m/s, and we attribute this phenomenon to the same merge degree in both cases.

Finally, Fig. 21 shows the average per-hop-delay experienced by flood messages in each approach. No new observations are obtained from this figure.

5. Discussions

The main cost of the proposed approach is that, when the flood traffic load is light, the per-hop-delay of the proposed approach is greater than that of the original approach. To solve this problem, we can use a controlled-load approach to dynamically shorten the merge delay when the flood traffic load is light. Even if the per-hop-delay of merged flood messages is still greater than that of the original approach, the proposed approach is still useful for those situations where applications are not delay-sensitive but mobile devices are energy-limited.

To implement this approach, a node needs to learn and cache all other active nodes' distances to itself. Although this would place an extra requirement for a mobile device, the required memory space depends only on the number of active nodes, not the number of all nodes of the whole network. In addition, since the cost of memory (e.g., the flash memory used by PDA) keeps decreasing but the battery usage lifetime does not increase significantly over the past few years, needing extra memory is less an issue than short battery usage hours for mobile devices.

6. Future work

Two topics can be investigated in the future. The first one is the merge effect on the routing path selection. Delaying the flood of an AODV's RREQ message at a node may change which routes are discovered, due to the operation of route discovery. The second one is the effect of a non-zero BER on the merge scheme's performance gains. It is interesting to see how well the double-Send scheme can cope with the increased PER (packet error rate) of merged messages.

Field experiments are also desired as experimental results are more convincing than theoretical simulation results. However, because conducting such experiments need a huge amount of manpower, efforts, and expensive equipments, we leave it as future work.

7. Conclusions

In this paper, we propose an approach to reducing the energy consumption of mobile nodes consumed by flood traffic. Our simulation results show that the proposed approach is effective. In the four tested cases, the achieved energy savings for flood traffic are 71%, 49%, 67%, and 36%, respectively. In addition, the flood message delivery failure rates are reduced by a factor of 76, 34, 40, and 22, respectively. We expect that more significant savings can be achieved in larger networks.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. This research was supported in part by MOE Program for promoting Academic Excellence of Universities under the grant number 89-E-FA04-1-4 and 91-E-FA06-4-4, the Lee and MTI Center for Networking Research, NCTU, and the Institute of Applied Science and Engineering Research, Academia Sinica, Taiwan.

References

[1] Open Shortest Path First Routing Protocol, RFC 1247.

[2] Charles Perkins, Elizabeth Royer, Ad hoc on demand distance vector routing, in: Second IEEE Workshop on Mobile Computing Systems and Applications, February 1999.

[3] Dipanjan Chakraborty, Anupam Joshi, Yelena Yesha, Tim Finin, GSD: a novel group-based service discovery protocol for MANETS, in: Fourth IEEE Conference on Mobile and Wireless Communication Networks, Stockholm, Sweden, September 2002.

[4] Laura Marie Feeney, Martin Nilsson, Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, IEEE INFOCOM'01, Anchorage, USA, 22–26 April, 2001.

[5] Robert Castaneda, Samir Das, Mahesh K. Marina, Query localization techniques for on-demand routing protocols in ad hoc networks, ACM MobiCom'99, Seattle, Washington, USA, 17–19 August, 1999.

[6] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, Jang-Ping Sheu, The broadcast storm problem in a mobile ad hoc network, ACM MobiCom'99, Seattle, Washington, USA, 17–19 August, 1999.

[7] M.S. Corson, S. Papademetriou, P. Papadopoulos, V. Park, A. Qayyum, An Internet MANET Encapsulation Protocol (IMEP), Internet Draft, draft-ietf-manet-imep-spec-01.txt, 7 August, 1999.

[8] IEEE Computer Society LAN MAN Standards Committee, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-1999. The

Institute of Electrical and Electronics Engineering, New York, 1999.

[9] S.Y. Wang, C.L. Chou, C.H. Huang, C.C. Hwang, Z.M. Yang, C.C. Chiou, C.C. Lin, The design and implementation of the NCTUns 1.0 network simulator, Computer Networks, doi:10.1016/S1389-1286(03)00181-6, in press.

S.Y. Wang is an Assistant Professor of the Department of Computer Science and Information Engineering at National Chiao Tung University, Taiwan. He received his Ph.D. degree in computer science from Harvard University in 1999. His research interests include computer networks, network simulations, and operating systems. He is the author of the Harvard network simulator and the NCTUns 1.0 network simulator. Due to its unique advantages, the NCTUns 1.0 network simulator was selected as a research demonstration by ACM MobiCom'02 held on 9/23/2002. As of 1/1/2003, it has been downloaded by more than 500 organizations worldwide since its release on 11/1/2002. More information about these simulators is available at http://NSL.csie.nctu.edu.tw/nctuns.html.