Ad Hoc Networks 9 (2011) 788–798

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

R-Code: Network coding-based reliable broadcast in wireless mesh networks ☆

Zhenyu Yang *, Ming Li, Wenjing Lou

Dept. of ECE, Worcester Polytechnic Institute, Worcester, MA 01609, United States

ARTICLE INFO

Article history:
Received 8 October 2009
Received in revised form 17 June 2010
Accepted 8 September 2010
Available online 29 September 2010

Keywords:
Wireless mesh networks
Network coding
Reliable broadcast

ABSTRACT

Broadcast is an important communication primitive in wireless mesh networks (WMNs). Applications like network-wide software updates require reliable broadcast to ensure that every node in the network receives the information completely and correctly. With underlying unreliable wireless links, a key challenge in implementing reliable broadcast in WMNs is to achieve 100% information reception rate at every node with high communication efficiency and low latency. Recently, network coding has emerged as a promising coding scheme in terms of communication efficiency especially for one to many communication patterns. In this paper, we put forward R-Code, a network coding-based reliable broadcast protocol. We introduce a guardian–ward relationship between neighboring nodes that effectively distributes the responsibility of reliable information delivery – from the global responsibility of the source to the localized responsibilities of guardians to their corresponding wards. We use a link quality-based minimum spanning tree as a backbone to guide the selection of guardians adaptively and the transmission of coded packets accordingly. Opportunistic overhearing is also utilized to improve the performance of the protocol. Extensive simulation results show that R-Code achieves 100% packet delivery ratio (PDR), while enjoying significantly less transmission overhead and shorter broadcast latency, compared with a state-of-the-art reliable broadcast protocol, AdapCode.

1. Introduction

Wireless mesh network (WMN) emerges as a promising technique to provide high-bandwidth Internet access to a large number of mobile devices in a specific area. Broadcast is an important function in WMNs. For example, it is necessary for software updates which may happen at the initial deployment and maintenance period, or is used in multimedia services like video/audio downloading. A key requirement of these applications is to strictly guarantee 100% packet delivery ratio (PDR), which means every node has to download every bit of the broadcasted file. In addition, efficiency is another important concern. Since other normal unicast traffics may exist in the network at any time, broadcast applications should have good coexistence with these traffics, which translates into consuming minimal amount of network bandwidth and disseminating the file with low latency.

It is nontrivial to design an efficient reliable broadcast protocols for real WMNs. The fundamental challenge comes from the unreliable nature of the wireless link [2], which is due to packet losses caused by channel fading and interferences. In order to guarantee 100% PDR with those unreliable links, some previous schemes [3,4] use automatic repeat request (ARQ) technique, which requires the receivers to provide explicit feedbacks of the packet reception status to the source. However, this will cause "ACK implosion" problem which may incur a large amount of redundant transmissions. Other schemes [5–7] combine ARQ with forward error correction (FEC) technique to reduce the transmission overhead while still guaranteeing

☆ The preliminary version of this paper appeared in Globecom 2009 [1].

* Corresponding author. Tel.: +1 508 831 5649.
E-mail addresses: zyyang@ece.wpi.edu (Z. Yang), mingli@ece.wpi.edu (M. Li), wjlou@ece.wpi.edu (W. Lou).

1570-8705/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2010.09.009

100% PDR. Yet, these technique consider the wireless link as point-to-point, and neglect the fact that wireless medium is broadcast in nature. This leads to duplicate transmissions at intermediate nodes, which are not efficient enough.

Recently, network coding (NC) has been proposed as an effective technique to increase the network bandwidth efficiency [8]. In contrast to FEC, NC gives intermediate nodes the ability of randomly encoding different packets received previously into one output packet. Thus, although multiple intermediate nodes may receive the same packet, they will broadcast different re-coded packets that are linearly independent with each other with high probability. Each of these re-coded packets can benefit other nodes that overhear it, which avoids the duplicate transmissions. Theoretical analysis has demonstrated that NC is able to approach the multicast and broadcast capacity in multi-hop wireless networks [9,10]. Due to the high complexity of implementing network coding, practical NC-based broadcast schemes have also been proposed [11–16], where NC is shown to have a noticeable gain in bandwidth efficiency. However, most of these schemes only provide reliability with best effort rather than guaranteeing 100% PDR, with the exception of AdapCode [13] and Pacifier [16]. However, AdapCode is purposefully designed for wireless sensor networks and is not efficient when directly applied into WMNs. While MORE and Pacifier focuses on multicast in WMNs, they do not consider to exploit the specific characteristic of broadcast, that is, every node has to receive the whole file. In fact, when a node has received a certain amount of information, it can be regarded as a temporary source, which can guarantee the reliable reception of its neighbors in an efficient way.

In this paper, we propose R-Code, an efficient distributed Reliable broadcast protocol in WMNs based on network Coding which guarantees 100% PDR. The core idea of R-Code is to establish a guardian–ward relationship between neighboring nodes, so that the global responsibility of the source to ensure the reliable reception of all the nodes in the network is distributed to all the guardians. This is because a guardian is a temporary source that is much closer to its wards than the original source, thereby it can guarantee their reliable reception of the file more efficiently. A link quality-based minimum spanning tree is constructed to guide the selection of guardians and packet transmissions accordingly. A guardian is the best node in a ward's neighborhood to ensure the reliable reception of the ward with the least number of transmissions. The guardian–ward relationship is adaptively maintained throughout the broadcast session to exploit the benefit of opportunistic overhearing. In addition, intra-flow NC is adopted to further reduce the total number of transmissions and simplify the coordination between multiple transmitters. Moreover, R-Code applies a source rate limiting mechanism to alleviate the collisions in the network. We evaluate R-Code and compare it with AdapCode by extensive simulations. The simulation results show that R-Code uses up to 15% less number of transmissions and 65% shorter broadcast latency than that in AdapCode to disseminate the same file.

The rest of the paper is organized as follows: we give related work in Section 2. In Section 3, we describe the preliminaries. The analysis of existing schemes is presented in Section 4. In Section 5, we introduce the design of R-Code protocol in detail. Section 6 presents the simulation results and Section 7 wraps up the paper.

2. Related work

Broadcast in multi-hop wireless networks has been studied for decades. From the perspective of reliability, those proposed schemes can be divided into two categories: (1) schemes that provide reliable broadcast services with best effort, a good survey of which can be found in [17]; (2) schemes that guarantee 100% PDR strictly. Some of them use ARQ technique [3,4], where the source requires feedbacks from all the receivers, leading to the well-known ACK implosion problem and also incurring large amount of redundant transmissions. Others combine FEC with ARQ [5–7], which can increase the throughput. However, all these schemes do not explicitly take advantage of the broadcast nature of wireless medium, and thus suffer from duplicate transmissions.

Recently, network coding has been applied to broadcast in multi-hop wireless networks. From the theoretical aspect, Lun et al. [9] prove that random linear network coding can be used to construct a capacity-approaching scheme for multicast over lossy wireless networks. Adjih et al. [10] show that by using a simple broadcast rate selection strategy, NC can ensure that every transmission is useful with high probability. Fragouli et al. [12] study NC-based efficient broadcast from both theoretical and practical point of views. They show that NC is able to increase the bandwidth/energy efficiency by a constant factor in fixed networks. They also propose a probabilistic forwarding-based algorithm for random networks which shows significant overhead improvement over probabilistic flooding. However, their algorithm does not guarantee 100% PDR.

From the aspect of protocol design, MORE [11] is the first practical NC-based routing protocol that achieves high throughput and guarantees 100% PDR, for both unicast and multicast sessions. The main idea of MORE is to combine opportunistic routing [18] with network coding, which eliminates the need of complicated coordination mechanism between multiple forwarders in pure opportunistic routing. However, MORE is inefficient when applied to multicast, since almost every node in the network may become a forwarding node (FN), which can cause heavy congestion [16]. Moreover, due to the constraint on the encoding/decoding complexity, the source has to divide the file into batches and transmit them sequentially. In particular, the source works in a stop-and-wait fashion, which means it needs to wait till a batch is received by all receivers before moving to the next batch. This makes MORE suffer from the "crying baby" problem [19]. Namely, when one receiver has poor connection to the source, trying to ensure 100% PDR of this receiver will make other receivers wait unnecessarily which can heavily degrade the performance of the whole protocol.

In order to address those weaknesses of MORE, Koutsonikolas et al. propose Pacifier [16], a high-throughput, reliable multicast protocol. In Pacifier, the source builds a shortest-ETX [18] tree, which is the union of all the shortest-ETX paths from the source to each receiver, to guide the multicast process. Since only non-leaf nodes of the tree are selected as FNs, the number of forwarding nodes is reduced significantly compared with MORE. Similar to MORE, Pacifier also applies intra-flow network coding and lets the source assign a *TX_credit* for each FN. This *TX_credit* is the expected number of transmissions this FN should make for each coded packet it receives from an upstream FN in order to ensure that each of its children receives at least one packet. To solve the crying baby problem, Pacifier lets the source send batches of packets in a round-robin way rather than the stop-and-wait fashion adopted in MORE. It also applies a source rate limiting mechanism to further reduce the congestion. However, both MORE and Pacifier are source routing protocols, which includes the list of FNs and their *TX_credit*s in the header of each transmitted packet. This makes them less scalable. Moreover, they do not exploit the specific characteristic of broadcast session and each receiver's reliable reception has to be guaranteed by the source through end-to-end ACKs, which is inefficient.

AdapCode [13] is a reliable broadcast protocol which is used for code updates in wireless sensor networks. Because we will compare R-Code against AdapCode in our evaluation, a brief overview of its two major components is given below.

2.1. Compressed forwarding

AdapCode works in a similar way like probabilistic forwarding. However, the forwarding is based on batch rather than single packet. That is, each node only transmits packets after receiving the whole batch. For a received batch which contains k packets, k/N coded packets is transmitted. The number N is called the "coding scheme", which is selected adaptively according to the number of neighbors. In this way, Adapcode reduces the number of transmissions significantly compared with pure flooding.

2.2. "Timer + NACK" mechanism

AdapCode ensures 100% PDR by a Timer + NACK mechanism, which runs as follows: each node i keeps a count-down negative ACK (NACK) timer and this timer will be restored to initial value once the node receives a packet. When the NACK timer counts to 0 and i still does not get enough packets for decoding, it broadcasts a NACK to request for the needed packets. Each of i's neighbors that overhears this NACK and possesses those required packets, is eligible to respond. In order to reduce the risk of unnecessary simultaneous responses, all those eligible responders will go through a coordination process. That is, each of them delays for a random period of time before responding to see if any other node is replying to this NACK. If no reply is heard during this period, it will respond to this NACK by sending all the requested packets. AdapCode also adopts a "lazy NACK" mechanism to reduce the number of NACKs, which requires each node to reset its NACK timer to avoid sending duplicate NACK if it overhears another one.

3. Preliminaries

3.1. Network model

The WMN considered in this paper consists of a number of wireless mesh routers that communicate with each other by radio transmission. Those mesh routers are static but not energy limited. The WMN connects to the Internet through some gateway routers. The broadcast application is one-to-all, where a gateway router is always the source that wants to disseminate a file to all the other routers in the WMN. The WMN is modelled as a weighted undirected graph $G(V, E)$, where V is the set of nodes (mesh routers) and E is the set of links. Two nodes i and j are considered to be connected if $w_{i,j} \geqslant w_{threshold}$, where $w_{i,j}$ is the weight of link (i, j) and $w_{threshold}$ is a given threshold value. $w_{i,j}$ is defined as the expected transmission count (ETX) [18] between i and j.[1] We also assume that for one transmission, the packet losses in different receivers are independent [20].

3.2. Network coding

We use intra-flow random linear network coding in this paper. In order to reduce the packet header overhead and encoding/decoding complexity, the source divides the broadcast file into small batches and send them sequentially. Each batch contains k original packets, denoted as p_i, $i = 1,2,\ldots,k$. The source keeps broadcasting coded packet of the current batch until all the receivers decode this batch successfully, then it moves to the next batch. We choose k to be 32 in our scheme, which is the same as in [11,16]. Each coded packet x is a linear combination of all the packets in the batch: $x = \sum_{j=1}^{k} \alpha_j p_j$, where $\langle \alpha_j \rangle$ is the encoding vector. Each α_j is randomly selected from a Galois Field $GF(2^q)$. We choose q to be 8 in our scheme, the same as in [11,16]. Every coded packet in transmission includes the encoding vector in the packet header. When a node receives a packet, it checks if the encoding vector of this packet is linearly dependent with all the other encoding vectors of the packets received previously. If so, this packet is discarded since the information it carries can be deduced from those already received packets; otherwise, this packet is called an innovative packet and stored in a buffer. When transmission opportunities appear, an intermediate node broadcasts coded packets generated by the packets stored in the buffer currently. Once a node receives k such innovative packets, it can decode this batch to retrieve all the original packets by doing Gaussian elimination, whose computational complexity is $O(k^3)$.

4. Existing schemes analysis

In this section, we carry out a thorough analysis of Pacifier and AdapCode.

[1] Each node will broadcast beacon messages every T seconds to estimate the link qualities to its neighbors.

4.1. Pacifier

In MORE and Pacifier, the selection of FNs and the calculation of credit for each FN are both based on ETX-metric. As stated in [21], this makes their performance heavily depends on the accuracy of the link quality measurement. Unfortunately, with currently widely used measurement mechanisms which are based on periodical hello packets, the precise link quality measurement can only be obtained with very high overhead, which is not applicable in practice.

Moreover, although Pacifier addressed the crying baby problem by letting the source send batches in a round-robin fashion, within the transmission of the same batch, it still suffers from the "ACK delay" problem [14]. That is, during the time between the ACK is sent from the receiver and it is received by the source, those coded packets sent by the source for the current batch are, in fact, unnecessary.

To solve this problem, Pacifier lets the source maintain a counter C_{S_i} for each batch i. C_{S_i} is the expected number of packets the source need to send to guarantee that at least one of the receivers receives the whole batch successfully. C_{S_i} decreases by one after each of the source's transmission. The source moves to the next batch either when it receives an ACK or when C_{S_i} reaches zero. However, since the calculation of C_{S_i} is based on the link quality measurement, the performance of this mechanism is also quite sensitive to the accuracy of those estimations.

We can see that the ACK delay problem is caused by the requirement of end-to-end ACKs (from the receiver to the original source), which can be further ascribed to that, in Pacifier, the source has to directly guarantee the reliable receptions of every node. In detail, since all the selected FNs in Pacifier only act as forwarders, which passively relay what they received from upstream nodes. For a given node i that still needs more packets, only the original source can actively inject the required packets into the network which then are relayed by those FNs to the receiver. Thereby the original source has to take the responsibility of ensuring 100% PDR for all the nodes.

However, since in broadcast, every node has to receive the whole file reliably sooner or later, a FN can also play the role of a temporary source after receiving the whole file. Thus for a given node i that still needs more packets, if some neighbor of i is a temporary source, i can get packets more efficiently from this temporary source than from the original source. In Section 5 we will show that by taking advantage of those temporary sources, we can design a reliable broadcast scheme which not only avoids ACK delay problem, but also does not require accurate link quality measurement.

4.2. AdapCode

In AdapCode, each eligible responder has the responsibility of replying to the overheard NACK by sending those required packets. In order to avoid that some nodes are too heavily loaded to die out quickly, which can potentially disconnect the wireless sensor network, AdapCode designs a random selection mechanism for the NACK's responder which makes every eligible responder has the same probability to reply. However, we argue that this random selection mechanism makes AdapCode inefficient for WMNs. This could be illustrated by a simple example.

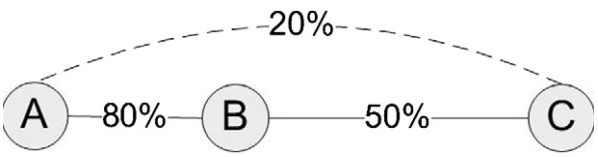


Fig. 1. A simple example of AdapCode. The packet delivery probability of each link is given.

Fig. 1 presents a scenario of local broadcast process, where node A just decoded a batch and tries to disseminate it to its neighbors. For convenience, we let the batch size to be 10 and the coding schemes of all the nodes to be 1, which means A will transmit 10 coded packets for this batch. Thus, B, C will receive 8 and 2 packets,[2] respectively. Since C has poorer connection to A, it will receive a packet with longer time interval expectedly which causes its NACK timer to fire earlier. Thus, it firstly sends out NACK, which in turn suppresses B's NACK because of the lazy NACK mechanism. In this NACK, C indicates the requirement of 8 more packets for successful decoding. Node A, as the only [3] eligible responder now, replies to this NACK by transmitting 8 packets. Because of the broadcast nature of wireless transmissions, B can also overhear these packets and obtain enough packets to decode the batch successfully. After decoding the whole batch, B will broadcast 10 coded packets according to the coding scheme, from which C can overhear 5 more packets. Now C has received $2 + 8 \times 0.2 + 10 \times 0.5 = 8.6$ packets, which is still not able to decode the batch. When its NACK timer fires again and sends another NACK, both A and B are eligible responders now. The total number of transmissions of this broadcast process is 32.9.

However, since the link quality of (B, C) is better than that of (A, C), if we let A cover [4] B at first and then B cover C deterministically, the total number of transmissions required is 27.5, which is more efficient. Since in practice, a broadcast session will experience similar situations quite often, we explicitly take the link quality into consideration in the design of R-Code.

5. R-Code design

5.1. Idea

The basic idea of R-Code is to distribute the responsibility of reliable information delivery from the original source to some selected nodes, called guardians. A guardian selects several nodes from its neighbors, called wards, and ensures the reliable reception of those wards. In order to

[2] The calculation is based on expected values in all the examples in this paper.

[3] The probability that B coincidentally possesses all the 8 required packets is quite small, we ignore it here for the convenience of analysis.

[4] A covers B means A has the responsibility of ensuring B's reliable reception.

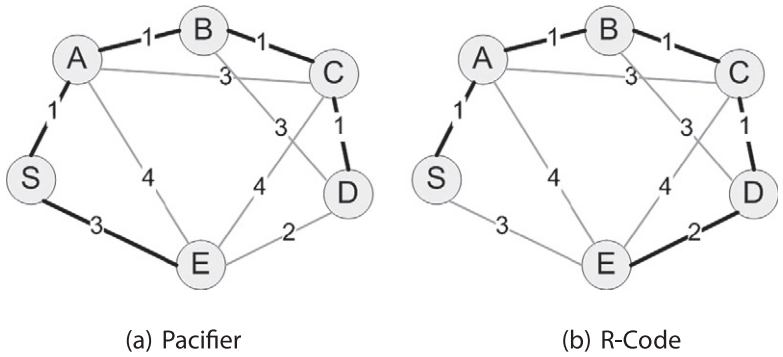


Fig. 2. A simple example to show the intuition of R-Code. The weight of each link is given.

promise that every node could be covered by a local guardian efficiently, R-Code uses a link quality-based minimum spanning tree as a backbone to guide the selection of guardians and wards.

In particular, the parent–child relationship between the node pairs in the tree can be translated into guardian–ward relationship. Therefore the original source only covers its children by keeping sending packets until get positive ACKs from all of these children. Upon receiving packet, each non-leaf child sends independently coded packets. A child stops sending only after receiving positive ACKs from all its children. This hop-by-hop cover process goes on until all the nodes receive this whole file reliably. Since these guardian–ward relationships are based on MST, which ensure that each node (except for the original source) will be assigned unique guardian, 100% PDR for all receivers are guaranteed. Also since all the ACKs only travel one hop distance, the ACK delay problem is avoided. Moreover, the optimal property of MST ensures each ward can always choose the best neighbor to be its guardian, avoiding the inefficiency of AdapCode. In addition, since R-Code defines specific guardian–ward relationship between neighboring node, it can use positive ACK rather than NACK adopted in AdapCode, which can also reduce the broadcast latency.

We use an example to explain the intuition underlying R-Code and compare the performance of R-Code with Pacifier. Without loss of generality, we assume the time is divided into slots and each transmission only happens at the beginning of a slot. Since every node has to send an ACK after receiving the packet, we ignore the cost of those ACKs for both protocols, which does not affect the comparison results. The broadcast latency is the number of slots from the time when the packet is broadcast firstly by the source to the time when it is received by the receiver.[5]

Fig. 2 presents a network, which consists of 6 nodes. *S* is the source and wants to broadcast a batch reliably, the size of which is *k*. If we build the broadcast tree like Pacifier does, which is to combine all the best unicast paths from source to every other nodes, then we obtain the tree that is shown in Fig. 2a with bold lines. The numbers of transmissions generated by every node are shown in Table 1 and the average number of transmissions is $0.91k$. However, we observe that *E* is not covered by its best neighbor. For example, it can get a packet from *D* with two transmissions, which is more efficient than getting it directly from the source *S*. In a comparison, MST can make each node to be covered by its best neighbor, which is shown in Fig. 2b with bold lines. Now the average number of transmissions needed is $0.65k$, which is reduced by almost 30%.

However, this gain comes with cost in average broadcast latency, which increases about 7%, from 2.02 to 2.17. This tradeoff between the number of transmissions and broadcast latency reflects the difference of goals between R-Code and Pacifier. That is, the primary goal of R-Code is to minimize the total number of transmissions while that of Pacifier is to achieve higher throughput, which, for a give file, can be translated into shorter broadcast latency. For this reason, we will not compare them in the evaluation part.

5.2. Design

R-Code works on top of the IP layer and the packet header format is shown in Fig. 3, which contains a type field that identifies data packet from control packet, the source's IP address, broadcast session id, batch index, the total number of batches for the file and code vector, which exists only in data packet and indicates the coefficients based on which the coded packet is generated from the original packets in this batch.

Table 1
Comparison between pacifier and R-Code.

Node ID	Number of transmissions		Broadcast latency	
	Pacifier	R-Code	Pacifier	R-Code
S	$3k$	$1k$	0	0
A	$1k$	$1k$	1	1
B	$2/3k$	$2/3k$	2	2
C	$7/9k$	$7/9k$	8/3	8/3
D	0	$4/9k$	31/9	31/9
E	0	0	3	35/9
Avg.	$0.91k$	$0.65k$	2.02	2.17

[5] For the simplicity of analysis, we calculate the broadcast delay for one receiver at a time.

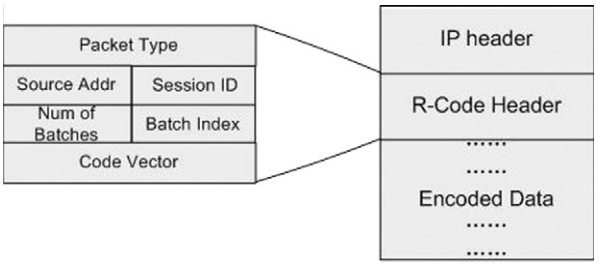


Fig. 3. R-Code packet header format.

5.2.1. Basic scheme

Generally, R-code can be divided into two stages. (1) Initialization. During this stage, a distributed algorithm [22] is applied to construct a MST.[6] During the whole broadcast session, this MST is considered to be fixed. As stated in Section 3.2, the source divides the file into batches with size of k and broadcasts those batches in a round-robin fashion to avoid crying baby problem.

For each batch, each node i initially selects its parent in the MST as guardian, denoted as G_i. As the broadcast session goes on, for some specific parent–child pair, their guardian–ward relationship maybe reversed.[7] However, we note that each node always has only one guardian at any time, either its parent or its child.

(2) Broadcast. During this stage, the source keeps sending coded packets generated from the current batch. When a node i receives a packet, it firstly checks whether this packet is innovative. If not, the packet is discarded; otherwise, i buffers this innovative packet and runs Gaussian elimination to check if it has gathered enough packets for decoding the whole batch. If not, it continues to keep silent and waits for more packets; else if i decodes the whole batch, it sends an ACK back to its guardian G_i by unicast. Then, if i is currently not in the process of disseminating some other previously decoded batch, it begins to play the role of guardian for its wards and keeps sending coded packets of this batch. Each guardian also works in a round-robin fashion to disseminate those successfully decoded batches received currently. After receiving ACKs from all the wards for a specific batch, the guardian will eliminate this batch from its buffer.

5.2.2. Dynamically maintain the guardian–ward relationship

In the initialization stage, selecting the parent to be guardian for its children is based on the expectation that the parent is closer to the original source and thus will receive the whole batch ahead of the children. However, since a node i can overhear packets not only from parent, but also from ancestors and siblings due to the broadcast nature of wireless transmissions, it could happen that i gets the whole batch successfully before its parent. In this case, if w_{i,G_i} is less than $w_{G_i,G_{G_i}}$, which means that node G_i can be covered by i with less number of transmissions than be covered by G_{G_i}, the better choice for G_i is to take i as the new guardian. We adaptively adjust the guardian–ward relationship to capture this opportunistic overhearing gain. This can be done by sending two notification packets to i and G_{G_i} separately by unicast. Otherwise, if w_{i,G_i} is greater than $w_{G_i,G_{G_i}}$, the guardian–ward relationship between G_{G_i} and G_i keeps untouched.

A simple example is shown in Fig. 4 to illustrate this adjusting process.The network consists of four nodes and S is the source. The MST is indicated by bold links in Fig. 4a. In the initialization stage, each node is assigned a guardian, G_A is S, G_B is A and G_C is B, as illustrated by arrows in Fig. 4b. Suppose the batch size is k. If the guardian–ward relationship is fixed, both S and A need to transmit $3k$ packets to broadcast this batch, totally $6k$ number of transmissions. However, we notice that after A's first k transmissions, C is expected to receive the whole batch successfully. Since $w_{C,B} = 1/3 \times w_{A,B}$, then B chooses A as its new guardian, as shown in Fig. 4c. C needs to transmit another $2k$ times to finish the broadcast session. The total number of transmissions is $5k$, which is 17% less than keeping the guardian–ward relationship fixed. The extra cost for this change is only 2 control packets, from B to A and C separately. Since the common value for k is 32 or 64, it worths the effort to dynamically maintain the guardian–ward relationship.

Moreover, this dynamic mechanism makes the initialization of guardian–ward relationship less critical. In specific, even the .measuring of the link qualities are not very precise which leads to a sub-optimal MST being built at the initialization stage, R-Code can still keep high performance during the broadcasting process. In another word, the performance of R-Code is not sensitive to the accuracy of link quality measurement, in sharp contrast with Pacifier.

5.2.3. Source rate limiting

The importance of source rate limiting, through which the contention level of the network can be reduced, has already been shown in recent studies [23]. Pacifier applies a simple backpressure-based rate limiting by exploiting the broadcast nature of wireless transmissions. The basic idea is to let the source wait before sending the next packet until it overhears that the child has already forwarded the previous packet it sent. AdapCode's approach is to let the source wait for a given period of time $T_{batchInterval}$, after finishing the current batch and before starting the next batch. Moreover, each node has to backoff for a random period of time before transmitting each packet. This time is uniformly chosen between 10 ms and 74 ms.

However, the situation in R-Code is different. Because now besides the original source, other guardians also generate packets actively rather than passively forwarding. Thus, R-Code can be considered as a multiple source broadcast scheme, which is different from flow-based Pacifier and AdapCode. Those backpressure-based approaches, which is derived from flow theory, is not directly applicable.

R-Code applies a simple rate limiting approach to all the guardians in the network. For a specific guardian, it has to

[6] In R-Code, the MST only need to be built up for once and can be shared by multiple broadcast sessions, this is different from Pacifier whose multicast tree structure needs to be constructed for every multicast session and reconstructed during the session when some receiver receives one batch successfully.

[7] Parent–child relationship is defined for MST which is fixed during the life time of the tree structure. However, guardian–ward relationship is defined according to the proceed of the broadcast session, which will be dynamically adjusted.

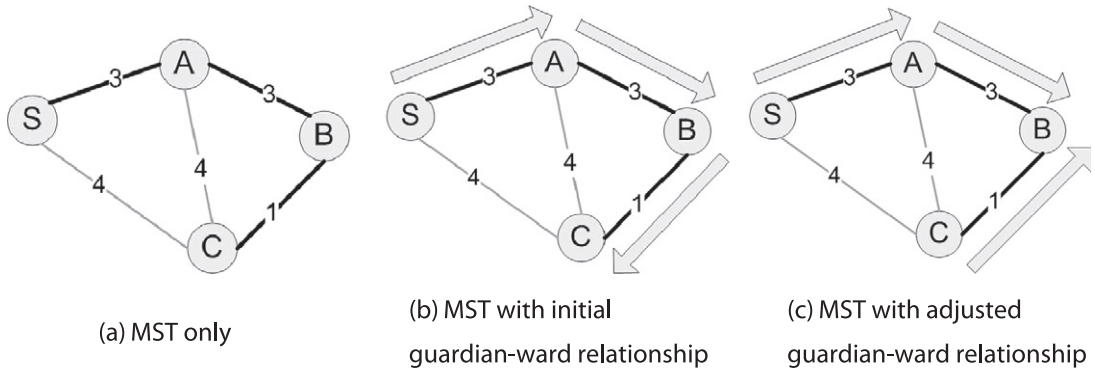


Fig. 4. A simple example of dynamically maintaining the guardian–ward relationship.

backoff a random period of time before each transmission, the range of which should be proportional to the number of concurrent transmitters this node knows in the neighborhood. For example, in an area with large number of concurrent transmitters, the waiting time should be longer in order to reduce the probability of collision; otherwise each transmitter waits for shorter time for fast dissemination. We apply the moving weighted average (MWA) [24] method to determine the average number of transmitters: $avgTransmitter = \alpha \times avgTransmitter + (1 - \alpha) \times curTransmitter$, where the *curTransmitter* is the number of concurrent transmitters that the node knows during the previous hello packet interval. The value of α should be determined according to the traffic stability of the network. Since most of the nodes in R-Code will be guardian for some batch during the whole broadcast session and the topology of WMN is stable, the number of transmitters among the neighbors of each node does not change often. Thus, we choose $\alpha = 0.8$ in our simulations. The wait period of time is uniformly chosen between $T_{pktInterval} \times 0.5 \times avgTransmitter$ and $T_{pktInterval} \times avgTransmitter$. $T_{pktInterval}$ is called packet broadcast interval, the value of which should be determined by the current traffic load in the network. We will evaluate the performance of various $T_{pktInterval}$s in the following section. Note that this approach is different from what is adopted by AdapCode, where the chosen range of the random waiting time is identical to all the nodes and the local environment is not taken into consideration.

6. Performance evaluation

We evaluate the performance of R-Code and compare it with AdapCode through extensive simulations. Since AdapCode is purposefully designed for wireless sensor networks, for fairness, we slightly modify it to be applied in WMNs. (1) In our implementation of AdapCode, we let the nodes transmit coded packets generated by linear combination of the whole batch rather than a portion of it. As claimed in [13], this could make AdapCode have better performance on bandwidth efficiency with the cost of higher computation overhead for encoding and decoding. (2) We let the responder of NACK send coded rather than original packets, which can benefits other nodes which overhear these packets, besides the sender of the NACK. (3) The range of the random backoff time is selected the same as R-Code does, which is more adaptive. (4) We also choose other parameters like batch size, the Galois field size, etc., to be the same as R-Code. Note that all those modifications make AdapCode performs better in WMN than its original version.

6.1. Simulation settings

We use Glomosim simulator in our simulation. The network consists of a 7×7 grid of static nodes. We choose the grid size to be 200 m and 250 m, where the former simulates a relatively dense network and the latter simulates a relatively sparse network. The average radio transmission range is 317m. Note that the WMN we considered in this paper consists of only routers, which are usually deployed in a well-considered way for the purpose of balancing the network coverage and deployment cost, so a grid topology is more reasonable than a random topology for simulation. For AdapCode, We follow the optimal coding schemes presented in [13], which is shown in Table 2.

The source is fixed to be node 0 for all the simulations. The broadcasted file is 4 MB, consisting of 1KB packets. Other related simulation parameters are listed in Table 3. We run both protocols in the two grid sizes 10 times and show the average results over all 10 runs. For each run, the $T_{pktInterval}$ ranges from 2 ms to 9 ms, with the step equals to 0.5 ms. We use the following metrics for comparison:

6.1.1. Average broadcast latency

the total time required for a node to receive the whole file, averaged over all nodes.

6.1.2. Average number of transmissions

the total number of transmissions of all the nodes divided by the number of nodes.

Table 2
Optimal coding schemes.

Average number of neighbors	0–5	5–8	8–11	11–
Best coding scheme	$N = 1$	$N = 2$	$N = 4$	$N = 8$

Table 3
Simulation parameters.

Simulation parameter	Value
Batch size	32
Galois fied size	2^8
$w_{threshold}$	5
$T_{batchInterval}$	100 ms
Data transmission rate	11 Mbps
ACK transmission rate	1 Mbps
Retry limit	7
Pathloss model	Two-ray
Fading mode	Rician
Rician k factor	4
Hello packet interval (T)	1 s

6.1.3. Average number of collisions

the total number of collisions experienced by all the nodes divided by the number of nodes. Note that one transmission can cause several collisions at different nodes.

6.1.4. Average number of linearly dependent packets

the total number of linearly dependent packets received by all the nodes divided by the number of nodes. Note that the count of this metric includes the case that a node who has already received the whole batch overhears coded packet from the same batch.

We do not compare PDR performance, since both R-Code and AdapCode can guarantee 100% reliability.

6.2. Number of transmissions

We first compare the average number of transmissions introduced by both protocols. Besides data packets, we also count the NACKS of AdapCode, and ACKs and those control packets for maintaining the guardian–ward relationships of R-Code.

From Figs. 5a and 6a we can see that whatever the grid size is, R-Code uses less number of transmissions than AdapCode does for accomplishing the broadcast session, the maximum reduction can be 13% when the grid size is 200 m and 15% when the grid size is 250 m. The key reason for R-Code's better performance is its local optimal decision. For each node, it always chooses the best neighbor to be guardian. In comparison, when a node i in AdapCode sends a NACK, it randomly chooses a node from those that overhears this NACK and possesses the required packets to be responder. This randomly selected node, as we argued in Section 4.2, maybe not the best one and thus incurs more redundant transmissions. This is shown clearly in Fig. 5b and 6b, where we can observe that AdapCode yields more linearly independent receptions in most cases.

The only exception appears when $T_{pktInterval}$ is small, e.g., less than 3.5 ms when the grid size is 200 m, and 2.5 ms when the grid size is 250 m, where R-Code uses more transmissions. The reason for this exception is that in these cases, all the nodes inject packets too fast to be sustained by the network. This causes heavy contention between

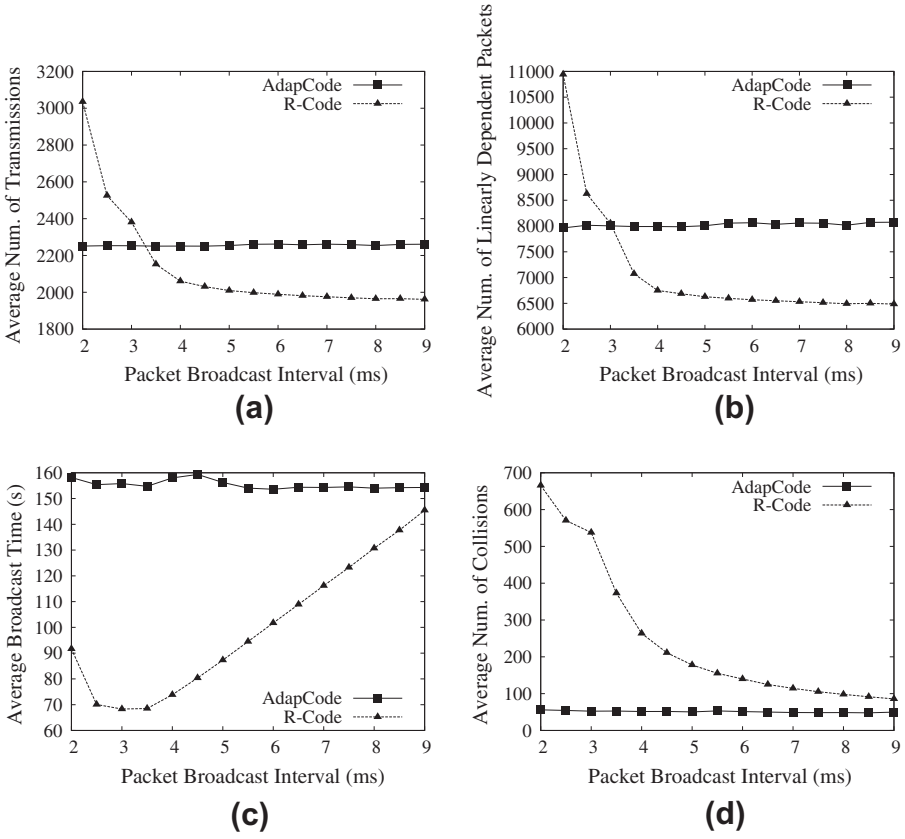


Fig. 5. Grid size = 200 m.

nodes, which is shown clearly in Fig. 6d, where we can see that the average number of collisions experienced by each node is almost 700 when $T_{pktInterval}$ is 2.0 ms and grid size is 200 m. The heavy collision further leads to the large amount of overheard linearly dependent packets, in other words, useless for the receiving nodes. This is shown in Fig. 6b. Since the average number of neighbors for each node is approximately six for the network with grid size of 200 m, when $T_{pktInterval}$ is 3.0 ms, all the nodes' broadcasting rate has already above 400 Kbps, which is quite a high speed. Note that the reason for AdapCode's better performance in these cases is the NACK + Timer mechanism, which makes AdapCode not sensitive to the variation of $T_{pktInterval}$. However, this comes with the cost of much longer broadcast delay, which is shown in Fig. 5c and 6c.

6.3. Broadcast latency

Compared with the performance of number of transmissions, R-Code gains larger advantage over AdapCode when considering the broadcast latency. We can see that under all settings, the average broadcast latency of R-Code performance better than AdapCode. When $T_{pktInterval}$ is small, the reduction ratio can be up to 65%, which is achieved when the grid size is 250 m and $T_{pktInterval}$ is 3.0 ms. This is consistent with our analysis in Section 5.1, where we point out that NACK mechanism inherently tends to elongate the broadcast latency. We also observe that the broadcast latency of R-Code grows almost linearly to $T_{pktInterval}$. This is because that before trying to inject packets into the network, each guardian has to delay for a random short period of time that is proportional to $T_{pktInterval}$. We also observe that this linear property does not hold when $T_{pktInterval}$ is small, e.g., less than 3 ms when grid size is 200 m. In these cases, the broadcast latency of R-Code is even higher than the case when the packet broadcast latency is 3 ms, which is also the minimum broadcast latency for all the cases. This seemly abnormal performance can also be explained by the high congestion level in these cases. Under such high congestion level, each transmitter will encounter many collisions that causes it to backoff longer time, according to the CSMA/CA mechanism, which offsets the benefits of shorter $T_{pktInterval}$s. In contrast, AdapCode's performance is quite stable under various $T_{pktInterval}$s. The reason is that the broadcast latency of AdapCode is mainly decided by the initial value of the NACK timer for each node, which is set to be $2 \times T_{batchInterval}$ and not related to $T_{pktInterval}$.

Although the performance of R-Code on both number of transmissions and broadcast latency is better than AdapCode, we still can see that there is a tradeoff between transmission overhead and broadcast latency. The network designer need to choose proper values for parameters according to specific applications.

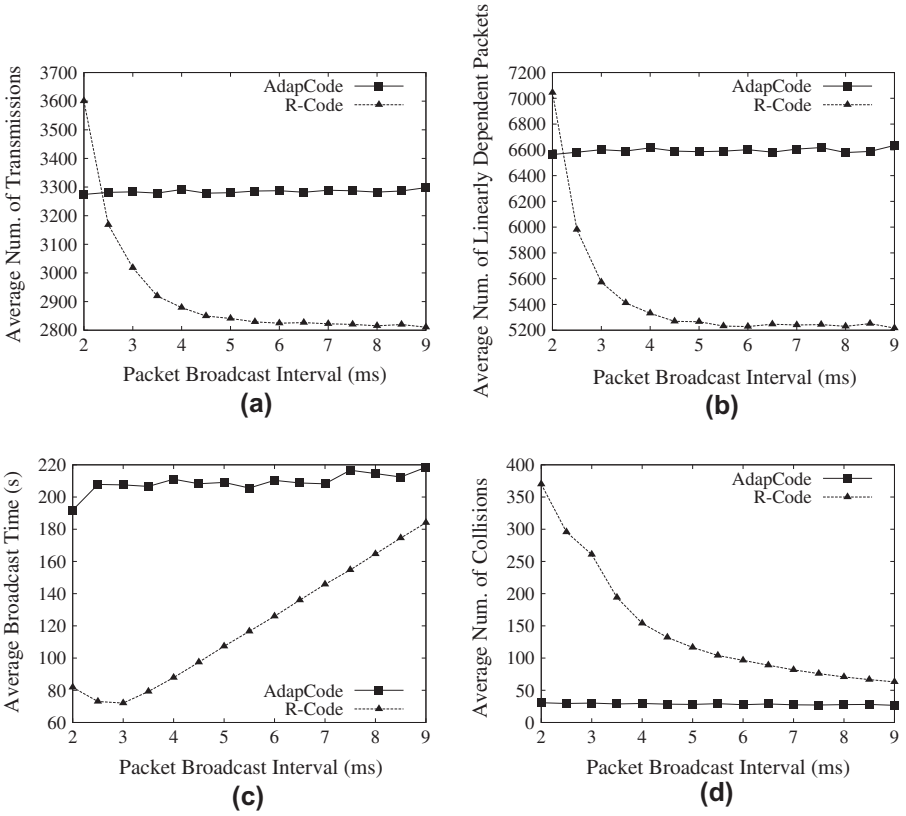


Fig. 6. Grid size = 250 m.

6.4. Discussion

We compare the average number of collisions between R-Code and AdapCode and observe that as the increasing of $T_{pktInterval}$, the average number of collisions experienced by AdapCode are quite stable. In a comparison, the average number of collisions experienced by R-Code decreased rapidly at first and then approaches to a stable value, which is a little higher than AdapCode. The reason for higher number of collisions experienced by R-Code in all settings is ''hidden terminal' problem. Since R-Code encourages simultaneous transmissions to enhance the performance of spatial reuse, so it tends to suffer from the hidden terminal problem more. On the opposite, the NACK mechanism of AdapCode makes each node's transmission more passive and provides less chance for the happening of hidden terminal problem. However, We argue that this does not means AdapCode will introduce less number of transmissions, because its inefficiency in bandwidth usage comes from the random selection mechanism for NACK's responder, which incurs large amount of linearly dependent packets. How to deal with hidden terminal problem in R-Code will be our future work.

7. Conclusion

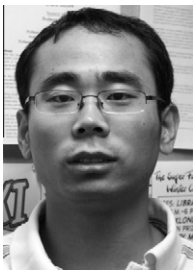
In this paper, we propose R-Code, a distributed and efficient broadcast protocol which guarantees 100% PDR for all receivers. By introducing a guardian–ward relationship between neighboring nodes, R-Code effectively distributes the global responsibility of reliable information delivery from the original source to those locally selected guardians. R-Code uses a link quality-based MST as a backbone to guide the selection of guardians adaptively and the transmission of coded packets accordingly. R-Code also prevent guardians from sending duplicated packets with no extra overhead by adopting network coding technique. Extensive simulations show that R-Code reduces the average number of transmissions and broadcast latency up to 15% and 65%, respectively, compared with AdapCode, a state-of-the-art reliable broadcast protocol under unreliable links.

Acknowledgement

This work was supported in part by the US National Science Foundation under Grants CNS-0746977, CNS-0716306, and CNS-0831628.

References

[1] Z. Yang, M. Li, W. Lou, R-Code: Network coding based reliable broadcast in wireless mesh networks with unreliable links, in: Globecom'09, 2009.
[2] N.Z. Marco Zú, B. Krishnamachari, An analysis of unreliability and asymmetry in low-power wireless links, ACM Trans. Sensor Networks 3 (2) (2007) 7.
[3] A. Sobeih, H. Baraka, A. Fahmy, ReMHoc: a reliable multicast protocol for wireless mobile multihop ad hoc networks, in: Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE, January 2004, pp. 146–151.
[4] E. Pagani, G.P. Rossi, Reliable broadcast in mobile multihop packet networks, in: MobiCom '97: Proceedings of the 3rd annual ACM/IEEE International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, 1997, pp. 34–42.
[5] L. Rizzo L. Vicisano, RMDP: an fec-based reliable multicast protocol for wireless environments, 1998.
[6] J. Nonnenmacher, E. Biersack, D. Towsley, Parity-based loss recovery for reliable multicast transmission, Networking, IEEE/ACM Trans 6 (4) (1998) 349–361. Aug.
[7] R.G. Kermode, Scoped hybrid automatic repeat request with forward error correction (sharqfec), SIGCOMM Comput. Commun. Rev. 28 (4) (1998) 278–289.
[8] C. Fragouli, J.-Y. LeBoudec, J. Widmer, Network coding: an instant primer, SIGCOMM Comput. Commun. Rev. 36 (1) (2006) 63–68. January.
[9] D.S. Lun, M. Medard, M. Effros, On coding for reliable communication over packet networks, IEEE Trans Inform Theory (2008).
[10] C. Adjih, S.Y. Cho, P. Jacquet, Near optimal broadcast with network coding in large sensor networks, in: First International Workshop on Information Theory for Sensor Networks, Sante Fe , USA, June 2007.
[11] S. Chachulski, M. Jennings, S. Katti, D. Katabi, Trading structure for randomness in wireless opportunistic routing, in: SIGCOMM '07, 2007.
[12] C. Fragouli, J. Widmer, J.-Y. LeBoudec, A network coding approach to energy efficient broadcasting: from theory to practice, in: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, April 2006, pp. 1–11.
[13] I.-H. Hou, Y.-E. Tsai, T. Abdelzaher, I. Gupta, AdapCode: adaptive network coding for code updates in wireless sensor networks, in: INFOCOM 2008, April 2008.
[14] L. Yunfeng, B. Li, L. Ben, CodeOR: Opportunistic routing in wireless mesh networks with segmented network coding, in: Network Protocols, 2008. ICNP 2008. IEEE International Conference on, October 2008, pp. 13–22.
[15] J.-S. Park, M. Gerla, D. Lun, Y. Yi, M. Medard, CodeCast: a network-coding-based ad hoc multicast protocol, Wireless Commun. IEEE 13 (5) (2006) 76–81. October.
[16] D. Koutsonikolas, Y.-C. Hu, C.-C. Wang, High-throughput, reliable multicast without crying babies in wireless mesh networks, in: CoNEXT. ACM, December 2008.
[17] B. Williams, T. Camp, Comparison of broadcasting techniques for mobile ad hoc networks, in: MobiHoc. ACM, 2002, pp. 194–205.
[18] D.S.J. DeCouto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, in: MobiCom '03: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, 2003, pp. 134–146.
[19] H. Holbrook, S.K. Singhal, D.R. Cheriton, Log-based receiver-reliable multicast for distributed interactive simulation, in: ACM SIGCOMM, 1995, pp. 328–341.
[20] A. Miu, H. Balakrishnan, C.E. Koksal, in: MobiCom '05: Proceedings of the 11th Annual International Conference on Mobile Computing and Networking, New York, NY, USA.
[21] D. Koutsonikolas, Y.-C. Hu, C.-C. Wang, ''CCACK: efficient network coding based opportunistic routing through cumulative coded acknowledgements, Tech. Rep. <http://web.ics.purdue.edu/~dkoutson/publications/ccack_techrpt.pdf>.
[22] R.G. Gallager, P.A. Humblet, P.M. Spira, A distributed algorithm for minimum-weight spanning trees, ACM Trans. Program. Lang. Syst. 1983.
[23] B. Hull, K. Jamieson, H. Balakrishnan, Mitigating congestion in wireless sensor networks, in: SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, ACM Press, New York, NY, USA, 2004, pp. 134–147.
[24] Y. lun Chou, Statistical Analysis, Holt International, 1975.



Zhenyu Yang (S'08) received his B.E and M.E degrees in Computer Science both from Xi'an Jiaotong University, China, in 2004 and 2007, respectively. He is currently a Ph.D. student in the Electrical and Computer Engineering department at Worcester Polytechnic Institute. His current research interests are in the area of wireless networks and network security, with emphases on network coding and protocol design.

Ming Li (S'08) received his B.E and M.E degrees in Electronic and Information Engineering both from Beihang University, China, in 2005 and 2008, respectively. He is currently a Ph.D. student in the Electrical and Computer Engineering department at Worcester Polytechnic Institute. His current research interests are in the area of wireless networks and pervasive computing, with emphases on protocol design, network and system security.

Wenjing Lou obtained her Ph.D degree in Electrical and Computer Engineering from University of Florida in 2003. She received the M.A.Sc degree from Nanyang Technological University, Singapore, in 1998, the M.E degree and the B.E degree in Computer Science and Engineering from Xi'an Jiaotong University, China, in 1996 and 1993 respectively. From December 1997 to July 1999, she worked as a Research Engineer at Network Technology Research Center, Nanyang Technological University. She joined the Electrical and Computer Engineering department at Worcester Polytechnic Institute in 2003 where she is now an assistant professor. Her current research interests are in the areas of ad hoc, sensor, and mesh networks, with emphases on network security and routing issues. She has been an editor for IEEE Transactions on Wireless Communications since 2007. She was named Joseph Samuel Satin Distinguished fellow in 2006 by WPI. She is a recipient of the U.S. National Science Foundation Faculty Early Career Development (CAREER) award in 2008.