R-Code: Network Coding based Reliable Broadcast in Wireless Mesh Networks with Unreliable Links

Zhenyu Yang, Ming Li and Wenjing Lou
Department of ECE, Worcester Polytechnic Institute, MA 01609
{zyyang, mingli, wjlou}@wpi.edu

***Abstract*— Broadcast is an important primitive in wireless mesh networks (WMNs). Applications like network-wide software update require reliable reception of the content with low-latency and high scalability (i.e., utilizing little bandwidth resource). In reality, the link layer broadcast transmission in WMNs is unreliable, which makes these goals hard to be attained at the same time. In this paper, we consider one-to-all broadcast scenarios and put forward R-Code, a reliable and efficient broadcast protocol based on intra-flow network coding. The key idea is to construct a minimum spanning tree as a backbone whose link weight is the expected number of transmissions on that link. The broadcast overhead and delay are simultaneously reduced by enabling the non-leaf nodes in the tree to move to the next batch of file segments as early as possible, while ensuring their downstream nodes reliably receive and correctly decode all the packets in the current batch. Opportunistic overhearing is utilized to further reduce the number of transmissions. Extensive simulation results show that our scheme always achieves 100% packet delivery ratio (PDR), while enjoying less broadcast overhead and much shorter delay than AdapCode, 14% and 50%, respectively.**

I. Introduction

Wireless mesh network (WMN) is envisioned to provide high-bandwidth Internet access for a specific area. It consists of some wireless mesh routers, which are static and not energy limited. One of them, called gateway node, connects the Internet and the WMN. In most cases, broadcast is an important function in WMNs. For example, it is necessary for software code updates which may be done at the initial deployment and testing phase of the network, or being used in multimedia services like video/audio downloading. Usually the gateway node gets file from remote ISP through Internet, and spreads this file to all the other routers within the network. The salient feature of such kind of applications is that, they require the PDR (Packet Delivery Ratio) to be strictly 100%, which means all the nodes have to download every bit of the broadcasting file. Also, since other normal unicast traffics may exist in the network at any time, broadcast applications are desired to have good coexistence with these traffics, which translates into consuming minimal amount of network bandwidth and propagating the file quickly.

In WMNs, the wireless links are unreliable, due to the channel fading [1]. This poses a great challenge to the design of highly reliable broadcast protocol with moderate overhead. Early broadcast schemes often incur large number of redundant transmissions for the purpose of reliability [2]. Moreover, the scheduling problem is also complicated by unreliable transmissions, since it is hard for the transmitting node to be synchronized on the receiving status of others, and this makes it difficult for the transmitting node to decide which packet should be sent next.

Network coding (NC) has been considered as an effective technique to increase the network bandwidth-efficiency in recent years. Briefly speaking, The essence of NC is to give nodes the flexibility of encoding different packets received previously together, for the purpose of benefiting multiple receivers with single transmission. With NC, the broadcast scheduling problem can also be eased in that the node does not need to concern about which packet should be transmitted out when MAC allows since different coded packets has the same probability to be useful for each neighbor. Many theoretical works have demonstrated NC is able to enhance the multicast and broadcast capacity [3], [4] in multi-hop wireless networks. Due to the high complexity of implementing network coding, practical and distributed NC-based broadcast schemes have also been proposed [5]–[8], and NC is shown to have a noticeable gain in increasing bandwidth-efficiency. However, most of them do not target at providing 100% reliability. While AdapCode [6] and Pacifier [9] do achieve this goal, we note that the former is purposely designed for wireless sensor networks and the latter is focus on multicast, which means both of them do not capture the specific characteristics of broadcasting in WMNs.

In this paper, we propose R-Code, a NC-based Reliable broadcast protocol in WMNs with unreliable links. Since we aim at the single source broadcast scenario, intra-flow network coding [10] is adopted to reduce the number of transmissions. The broadcasting file is chopped into small batches of packets, called generations [11], where received packets from one generation are randomly and linearly recombined (coded) and then rebroadcasted. In R-Code, In order to guarantee the perfect reliability with minimal overheads, we build a minimum spanning tree (MST) to play the role of a "virtual backbone", whose link weight is based on ETX metric. *The key idea of R-Code is that every node needs to have a "guardian" who is responsible for its reliable reception and correct decoding of a generation and for each node except the source, its guardian is the parent node in the MST*. Since each guardian node only needs to take care its children, whose link qualities are relatively better than its other neighbors and thus are expected to get the whole generation faster, its transmissions tend to be useful for all the neighbors and thus reduces the number of redundant broadcasts. 100% reliability is guarantee by requiring each child node to send a positive, generation level ACK back to the guardian. R-Code continues to exploit the broadcast nature of wireless transmission to further reduce the number of transmissions required. We note that this guardian-child relation is not fixed. The details will be shown in Section IV-C.

Another contribution of R-Code is that, the average broadcast delay is greatly reduced. since a guardian only need to

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

guard those best connected neighbors, it supposes to complete the current generation quickly and moves to the next generation fast. Simulation results show that, R-Code can guarantees 100% reliability, while the broadcast overhead is smaller than AdapCode. Meanwhile, R-Code yields much lower broadcast delay. These results show that R-Code is high-performance and practical for broadcast in WMNs.

The rest of the paper is organized as follows. We give related work in Section II. In Section III, we describe the network model and network coding primitives. In Section IV, we introduce the design of R-Code protocol in detail. In Section V, we present the simulation results. Section VI presents the future works and Section VII wraps up the paper.

II. Related Work

Broadcast in multi-hop wireless networks has been studied for years and a good survey of these proposed schemes can be found in [12]. Most of these schemes model the network as Unit Disk Graph (UDG), which suppose a binary status for the receivers: if the receiver stays within the disk-like transmission range, it can receive the transmitted packet, otherwise absolutely not. Although this assumption is simple and neat, it is not consistent with practical observations [13] where fading and interference effects make the reception probability to be an continuous value. Some protocols try to deal with this more practical model through ARQ and FEC techniques or a combination of them, such as Reliable Multicast Algorithm [14],etc. However, these mechanisms tend to incur high extra overhead and cause new problems like "ACK explosion", etc.

Network coding has been introduced into broadcast research only recently. On the theoretical aspect, D. S. Lun *et.al* [3] prove that random linear network coding is capacity achievable under lossy links by solving linear programming. Adjih et.al. [4] show that by using a simple broadcast rate selection strategy, network coding can asymptotically assure that every transmission is innovative for the receivers, however, they assume UDG model. Fragouli *et. al* [5] study NC-based efficient broadcast from both theoretical and practical point of view. 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, and show big improvement over probabilistic flooding. However, their work assumes every node has packet to send which is not applicable for one-to-all scenarios. Also it does not aim at providing perfect reliability and assumes UDG model.

On protocol design aspect, AdapCode [6] studies global code update in wireless sensor networks (WSNs) which guarantees perfect reliability. It also tries to achieve load balance and rapid propagation. Essentially, this scheme reduces the traffic by letting a node send one coded packet after receiving every *N* coded packets from other nodes, which is similar to traditional probabilistic flooding and it applies a "NACK+Timer" mechanism to promise the 100% reliability. However, since the NACK mechanism inherently tends to elongate the reception time of each generation, AdapCode still needs relatively long propagation delay. Also, the link quality is not explicitly considered in the design of AdapCode, which may result in additional transmissions in realistic networks.

The Pacifier [9] is a high-throughput multicast protocol which alleviates the "crying baby" problem. Namely, when one of the destinations has very poor connection and if we try to satisfy its reliability requirement, then the rest of the destinations will experience performance degradation. Pacifier combines intra-flow NC with tree-based opportunistic routing, and solves the "crying baby" problem by sending generations in a round-robbin way.

III. Preliminaries

A. Network Model

In this paper, the wireless mesh network considered only consists of all the mesh routers rather than the combination of routers and client users. one of these routers plays the role of gateway that connects to the Internet. The broadcast application is one-to-all, where the gateway node is always the source who wants to spread some file through the whole network. This WMN is modelled as an weighted undirected graph $G(V, E)$, where V is the set of nodes (mesh routers) and E is the set of links. The weight of link (i, j) is: $w_{i,j} = (1/p_{i,j} + 1/p_{j,i})/2$, where $p_{i,j}$ is the probability of successful packet reception from i to j, vice versa. Since in reality mesh routers are often fixed, we assume the network topology and also the link qualities keep stable during one broadcast session [15], which is usually finished within tens of seconds. The mesh routers broadcast beacon messages every T period of time, in order to maintain local neighborhood topology and measure the link qualities, which is calculated as the number of beacons received divided by the number of beacons generated in T. Moreover, we assume each node has enough memory that can store several generations simultaneously, where the length of each generation usually is tens of kilo-bytes.

B. Network Coding

We use intra-flow random linear network coding in this paper. The broadcasted file is divided into equal-length segments sequentially, each segment fits into one packet and every k packets together consist of a generation. Within one generation, the original packets is denoted as $s_i, i = 1, 2, ..., k$. A coded packet x is a linear combination of all these k original packets: $x = \sum_{j=1}^{k} \alpha_j s_j$, where $< \alpha_j >$ is the encoding vector, each element of which is randomly selected from a Galois field $GF(2^q)$. Every coded packet includes the coding vector in its packet header.

Upon receiving a coded packet x of generation i, the receiver adds x into the buffer matrix for generation i and then tries to decode this generation by doing Gaussian elimination on the buffer matrix, whose complexity is $O(k^3)$.

C. Minimum Spanning Tree

There are efficient distributed algorithms to calculate the MST [16] for a given network. In our protocol, since we assume the WMN is static, the MST can be computed and updated distributively along with the routing table at relatively long intervals, the extra communication and computation overheads introduced can be amortized by several broadcast sessions and thus be negligible.

978-1-4244-4148-8/09/$25.00 ©2009

Fig. 1. The packet format of R-Code

IV. R-CODE

The intuition of R-Code is to find a guardian G_i for each node i, which is the most suitable neighbor of i to promise the 100% reliable reception. The suitable here means G_i can transmit one packet to i reliably with minimum number of transmissions. We believe a good global performance can be achieved through all those simple, optimal local decisions. We note that our effort is put on designing practical and efficient protocol rather than pursing theoretically optimal performance.

A. Protocol Overview

R-code can be divided into two stages.

1) Initialization Stage: during this stage, each node periodically broadcast beacons to estimate the quality of links to its neighbors. Based on these estimations, each node i builds up a neighbor table $Table_i$, $Table_i = \{j | w_{i,j} \leq W_{threshold}, j \in V \text{and } j \neq i\}$, where $W_{threshold}$ is some predefined threshold value to eliminate those poor connected neighbors for the simplicity of calculation. Further, based on the content of $Table_i$, we continue apply a distributed algorithm [16] to build a minimum spanning tree. Node i stores this tree structure by a node set which contains all its neighbors within the MST.

Above is the general initialization stage. For some specific broadcast session, the single source s becomes root and makes the MST just built a directed tree originated from itself. Each node i records the upstream node as its Guardian $Guardian_i$, and all the downstream nodes as $Children_i$. The $Guardian_s$ is s itself and the children sets of all the leaf nodes is empty. s also spreads some basic information about the forthcoming broadcast session to all other nodes, which includes generation size G_{size} and total number of generations of this broadcast session G_{num}. These information will be used to judge the completion of this broadcast session.

2) Broadcast Stage: The broadcast session begins by keeping the source sending coded packets generated from the first generation with packet interval T_{pkt}. The packet format is shown in Fig.1. The value set of PKT_TYPE segment is $\{ack = 0, data = 1\}$, whcih tells the basic type of this packet. The value set of GUARDIAN_UPDATE segment is $\{normal = 0, guardianReq = 1, childReply = 2, deny = 3\}$, the meaning of which will be illustrated in the following section. GEN_INDEX is a short unsigned integer and tells which generation this packet belongs to.

When a node i overhears a packet, it runs Gaussian elimination to check if it received enough information to decode all packets within this generation. If the information received is still not enough, it keeps silent and waits; Else if i decode the whole generation, it records this reception time and notifies the successful decoding to its guardian by sending a positive ACK. Then, if i is currently not in the process of broadcasting for some other generation, it begins to play its role of guardian for its children and keeps sending coded packets with interval T_{pkt}.

In R-Code, guardian node needs to receive the ACKs from all its children before moving to the next generation. After receiving all those ACKs, the guardian node will deliver the packets of this generation to upper layer and flush its buffer. Additionally, it should add this generation's index into the successfully acknowledged ones and pause for a period of $T_{generation}$ time to allow children to rebroadcast the generation they just received.

The guardian selects the next generation to transmit like below: For source node, it just moves sequentially, from generation n to generation $n + 1$, until complete the last generation and then quit this broadcast session. We note that source's quit does not mean the terminate of the broadcast session, which may still go on in other nodes. For other guardians, the selection of next generation follows FIFO policy: From those successfully received but not successfully acknowledged generations, the guardian chooses the earliest received one, whose index is always the smallest too. if there are no such generations currently, then this node checks the previous records of successfully acknowledged generations to see if all the G_{num} generations have been acknowledged. If so, it quit this broadcast session; or else it keeps silent and waits for future reception.

B. Dealing with ACKs

As described above, the perfect reliability is guaranteed by the guardian-child relation. For specific generation, if the guardian node does not receive all the positive ACKs from its children, it will keep sending coded packet from this generation until all children are satisfied. Since multiple ACKs will be send back to the guardian, how to avoid introducing the notorious "ACK explosion" problem is important. We make a requirement about the generation size used in R-Code, which should be relatively large, i.e, 32 or 64. Because those ACKs is in generation-level rather than packet-level, if one generation contains large number of packets, thus for some guardian node, the link qualities between it to its children are different and those child nodes tend to receive the whole generation successfully at different times with very high probability. Further, we require that each child backoffs a random short period of time before sending ACK. Finally, since the children are a small subset of all the neighbors of the guardian, the number of which is usually no greater than 3, so even some of them really reply ACKs almost at the same time and cause a conflict at the guardian, the collision avoidance mechanism of lower MAC layer can easily handle this.

We note that the use of positive ACK can greatly reduce the average propagation delay compared with NACK, which is applied in AdapCode. Since NACK introduces extra waiting time for guardian, which, after sending predefined number of times, needs to wait for the replies from children to tell if more transmissions are needed. The reason for AdapCode to apply NACK is to achieve load balance between all nodes in the network. It is indeed an appropriate choice for sensor networks, where load-balancing is critical for prolonging the lifetime of the network and thus much more important than delay metric. However, the situation in WMNs is different, the routers are not constrained by energy supply, so broadcast protocols in WMNs regard average propagation delay, which also can be translated into throughput, as more important metric.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

C. Dynamically Maintaining Guard Relation

In R-Code, the MST just need to be built up for once and can be shared by many broadcast sessions, this is different from Pacifier whose multi-cast tree structure needs to be constructed for every multi-cast session and reconstructed during the session when some destination node finishes a generation. But for each specific broadcast session, more specifically, for each generation within this session, R-Code needs to change the guardian-child relations dynamically to capture the gain that comes from broadcast nature of wireless transmissions.

Since a node i can overhear packets not only from parent, but also from ancestors and siblings because of the broadcast nature of wireless transmissions, which actually transforms the tree structure into a mesh, so it could happen that the child gets the whole generation successfully before its guardian. In this case, if w_{i,G_i} is less than $w_{G_i,G_{G_i}}$, which means node G_i can be covered more efficiently by i than G_{G_i}, then the better choice for G_i is to take i as the new guardian. 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-child relation between G_{G_i} and G_i keeps the same.

In detail, if i never overheard G_i's transmission about this generation, whatever the ACK or coded packets, it considers that G_i still not receive this generation successfully, then for the ACK sending to G_i, i sets the GUARDIAN_UPDATE segment of the ACK packet to be 1, which means i ask for G_i to be its child. When G_i receives this ACK, if it actually has already got this generation, it replies with a packet whose PKT_TYPE=0 and GUARDIAN_UPDATE=3, which means it denies i's request; or else if it is indeed needs more information of this generation, then it compares the link weights $W_{G_i,i}$ and $W_{G_{G_i},G_i}$, if the former is smaller, it reply with a packet whose PKT_TYPE=0 and GUARDIAN_UPDATE=1, which means it accepts i as its new guardian; Or else it sets the GUARDIAN_UPDATE of this packet to be 3.

A simple example is shown in Fig.2. S is the source. Those bold links indicate the structure of MST. 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.2(b). Suppose the generation size is k. S and A need to transmit $2 \times k$ times both to broadcast this generation, totally $4 \times k$ number of transmissions. However, we notice that when A transmits $2/3 \times k$ times, actually C has already got the whole generation successfully. Since $w_{C,B}$ is half of $w_{A,B}$, then B replaces its guardian A with C, just like showed in Fig.2(c). And C needs to transmits another $2/3 \times k$. The extra cost for this update of the guardian-child relation between B and C is 2 unicasts, from B to A and C separately, so the total number of transmissions is $(2 + 4/3) \times k + 3$. Since one generation contains tens of packets generally, we can see that the effort of dynamically maintaining the guardian-child relation is deserved.

V. Performance Evaluation

We evaluate the performance of R-Code and compare it with AdapCode ,which performs the best in the state-of-the-art schemes, through extensive simulations. Actually in our implementation of AdapCode, for fairness, we always allow the nodes to transmit coded packets generated by linear combination of the whole generation rather than a portion of it. As claimed in [6], this could give adapCode better performance on bandwidth efficiency, but this also will increase the computation cost for encoding and decoding, so it is not proper for sensor networks. However, since the routers in WMNs are much powerful than sensors in WSNs, this modification will make adapCode performs better in WMNs. And we also choose other parameters like generation size, the Galois field size,etc, to be the same as R-Code. We don't compare with Pacifier, because it is purposely designed for reliable multicast session, The way it builds the tree structure, which is the union of all the shortest-ETX paths from the source to each destination, behaves obviously bad when extending to broadcast session [17].

TABLE I

OPTIMAL CODING SCHEMES

average neighbor	0-5	5-8	8-11	11-
N	1	2	4	8

TABLE II

SIMULATION PARAMETERS

Simulation Parameter	Value
$W_{threshold}$	5
$T_{generation}$	100ms
Random backoff time	10..30ms
data transmission rate	11Mbps
ACK transmission rate	1Mbps
Retry limit	7
pathloss model	two-ray
fading mode	rician
rician k factor	4
Hello packet interval(T)	1s

A. Simulation Settings

We use Glomosim simulator in our simulation. The network consists of a 7×7 grid of static nodes, where the grid size equals $200m, 250m$. The corresponding average number of neighbors per node are 9.51, 6.39, which means the first network is relatively dense and the latter is relatively sparse. We note that the WMN we considered in this paper consists of only routers, which are usually deployed in an well-considered way for the purpose of coverage and cost, so the density will be moderate. For AdapCode, We follow the optimal coding schemes presented in [6], which is shown in Table I.

The source is fixed to be node 0 for all the simulations. The broadcasted file is 1MB and being divided into 4096 pieces, each of which is 256 byte long. Other related simulation parameters are listed in Table II. We run both protocols in the two grid size 10 times with T_{pkt} range from $10ms$ to $30ms$. We use the following metrics for comparison:

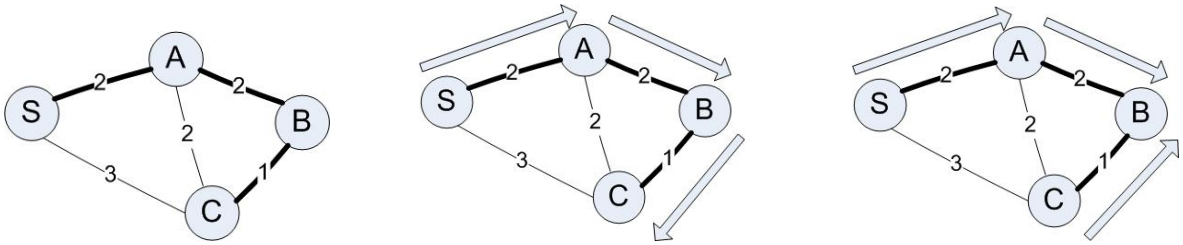
Average propagation delay: the total time required for a node to receive the whole file, averaged over all nodes.

Average number of transmissions: the total number of transmissions of all the nodes divided by the node number. It gives an estimate of the average traffic introduced by this broadcast session.

Average number of collisions: the total number of collisions of all the nodes divided by the node number.

Average number of linearly dependent packets: the total number of linearly dependent packets received by all the nodes divided by the node number. We note that the count of this

978-1-4244-4148-8/09/$25.00 ©2009



(a) MST only (b) MST with initial guard relation (c) MST with modified guard relation

Fig. 2. A simple example of dynamic guard relation

metric includes the case that some already full-ranked node overhears coded packet for the same generation.

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

B. Traffic

We first compare the average traffic introduced by both protocols. Besides data packets, we also count the NACKS of AdapCode and those unicast packets for maintaining the guardian-child relation and ACKs of R-Code. From Fig.3(a), Fig.4(a) we can see that whatever the gird size is, R-Code uses less number of transmissions than AdapCode does for completing the broadcast session, the maximum reduction can be 10% in sparse network and 14% in dense network. The only exception is appeared when gride size is $250m$ and packet broadcast interval is $11ms$, where R-Code uses more transmissions. The reason for this exception is that in this case, the source pumps packets too fast to be sustained by this network. This causes heavy contention between nodes, which is shown clearly in Fig.4(c), where we can see that the average number of collisions experienced by each node is almost 600. The heavy collision further leads to the large amount of packets overheard to be linearly dependent, in another word, useless for the receiving nodes. This is shown in Fig.4(b). Actually, when T_{pkt} is $11ms$, the source's broadcasting rate already reaches $186Kbps$, which is quite a high speed. We want to note that the reason for AdapCode's better performance under this situation is it's NACK+Timer mechanism, which makes AdapCode not sensitive to the variation of T_{pkt}, and this can easily be seen from Fig.3(a) and Fig.4(a).

However, If the source transmits packets with moderate rate that can be sustained by the network, R-Code performs quite better than AdapCode. The key reason is R-Code's local optimal decision. For each node, it always choosing the best neighbor to be guardian. In comparison, when a node i in AdapCode requires some more packets, it just randomly chooses a node from those who overheard i's NACK and also able to reply. This randomly selected node, we argue, maybe not the best one and thus needs more number of transmissions to satisfy the requirement of node i. Another observation can explain the second cause for the more transmissions of AdapCode. Since the timer of each node in AdapCode is restored to initial value once this node receives a new packet due to the "lazy NACK" mechanism, on the opposite, this means if the node does not receive any packet for some period, it will first fire its timer and broadcast the NACK. Unfortunately, this is just what happened to those nodes with bad connection to the sender. Further, because of the poor links, each of these nodes tends to receive much fewer packets when its timer fires, so it requires more retransmissions, most of which will be useless for those good connected receivers who only need small number of innovative packets. This is shown clearly in Fig.3(b) and Fig.4(b), where we can observe that AdapCode encounters more linearly independent receptions in most cases.

C. Propagation Delay

Compared with the performance of introduced traffic, R-Code gains larger advantage over AdapCode when considering the propagation delay. It is obviously to see that under all settings, the average propagation delay of R-Code performances much better than AdapCode. When T_{pkt} is small, the reduction ratio can be almost 50%. This is consistent with our analysis in previous section, where we point out that NACK mechanism inherently tends to elongate the propagation delay. We also observe that the propagation delays of both protocols grow almost linearly to the packet broadcast interval. This is because all the transmitters pump packets into network with a period of T_{pkt}. The relatively irregular performance of AdapCode in Fig.4(d) is due to the NACK mechanism again. Because the responder is chosen randomly, if it is a better one, then the propagation will be a little faster, otherwise the propagation is relatively slower. And since the network is sparse when grid size is $250m$, for some node i which send an NACK request, the number of potential responders is small and their distances to i, which correspond to the link qualities, also vary greatly and this tends to make the random select policy performance unstable. Actually, this instability is shown in Fig.4(a) and Fig.4(b). We can see that there is a tradeoff between transmission overhead and broadcast delay, so specific applications should choose proper setting for their own use.

VI. Future Works

We make a comparison of average number of collisions between R-Code and AdapCode and observe that as the increasing of T_{pkt}, the average number of collisions experienced by AdapCode only decreases a little, from 96 to 70 when grid size is 200m and 65 to 46 when grid size is 250m. While that of R-Code decreases 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 "hidden terminal" problem more. On the opposite, the NACK mechanism

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

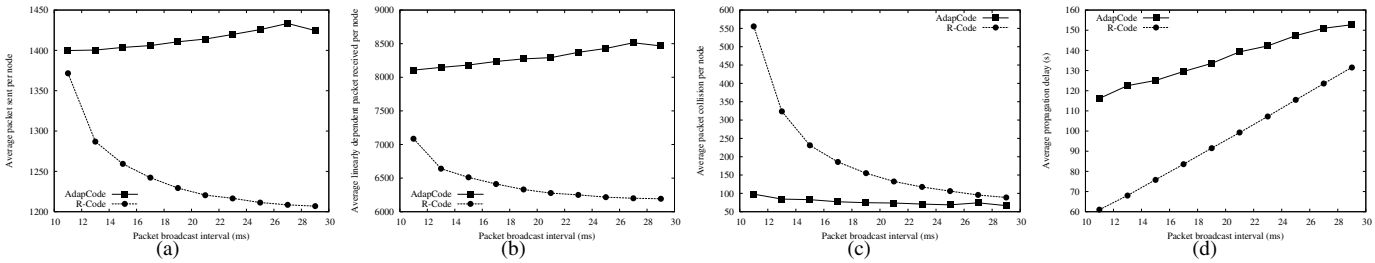


Fig. 3. Grid size=200m, Neighbor size=9.51

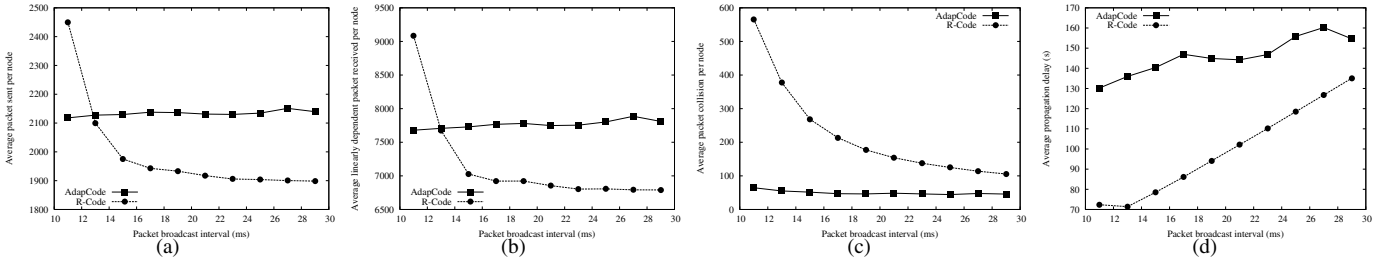


Fig. 4. Grid size=250m, Neighbor size=6.39

of AdapCode makes each node's transmission more passive and provides less chances for "hidden terminal" problem to happen. However, We argue that this does not means AdapCode will introduce less traffic, because its inefficiency in bandwidths usage comes from the random choosing policy for the responder, which incurs large amount of linearly dependent packets. What we interested here is that, since R-Code has higher average number of collisions, that means it has larger improvement space. How to deal with "Hidden Terminal" problem in R-Code will be our future work.

VII. Conclusions

In this paper, we focus on designing a practical broadcasting protocol which provides absolute reliability for all receivers. We present R-Code, A simple, efficient and high-performance broadcast protocol with the help of intra-flow network coding to reduce the average number of transmissions introduced and propagation delay. The core idea is to promise each node is covered by the most efficient neighbor. Based on the intuition that local optimal solution can achieves better global performance, R-Code builds a MST as "virtual backbone" to guide the broadcast process. Extensive simulations show that R-Code can reduce the average number of transmissions and propagation delay as high as 14% and 50%, respectively, compared with the state-of-the-art AdapCode.

VIII. Acknowledgements

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

References

[1] N. Z. Marco Zú and B. Krishnamachari, "An analysis of unreliability and asymmetry in low-power wireless links," *ACM Trans. Sen. Netw.*, vol. 3, no. 2, p. 7, 2007.

[2] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *IEEE/ACM MobiCom*, 1999, pp. 151–162.

[3] D. S. Lun, M. Medard, and M. Effros, "On coding for reliable communication over packet networks," *IEEE transactions on information theory*, 2008.

[4] C. Adjih, S. Y. Cho, and 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.

[5] C. Fragouli, J. Widmer, and J.-Y. Le Boudec, "A network coding approach to energy efficient broadcasting: From theory to practice," *INFOCOM 2006*, pp. 1–11, April 2006.

[6] I.-H. Hou, Y.-E. Tsai, T. Abdelzaher, and I. Gupta, "Adapcode: Adaptive network coding for code updates in wireless sensor networks," *INFOCOM 2008.*, April 2008.

[7] L. Yunfeng, B. Li, and L. Ben, "Codeor: Opportunistic routing in wireless mesh networks with segmented network coding," *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, pp. 13–22, Oct. 2008.

[8] J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Medard, "Codecast: a network-coding-based ad hoc multicast protocol," *Wireless Communications, IEEE*, vol. 13, no. 5, pp. 76–81, October 2006.

[9] D. Koutsonikolas, Y.-C. Hu, and C.-C. Wang, "High-throughput, reliable multicast without crying babies in wireless mesh networks," in *CoNEXT*. ACM, Dec 2008.

[10] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *SIGCOMM '07*, 2007.

[11] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, vol. 41, no. 1, 2003, pp. 40–49.

[12] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *MobiHoc*. ACM, 2002, pp. 194–205.

[13] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *SenSys '03*. New York, NY, USA: ACM Press, 2003, pp. 1–13.

[14] T. Gopalsamy, M. Singhal, D. Panda, and P. Sadayappan, "A reliable multicast algorithm for mobile ad hoc networks," in *22nd IEEE International Conference on Distributed Computing Systems*, July 2002, pp. 563–570.

[15] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-based models of delivery and interference in static wireless networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 51–62, October 2006.

[16] J. A. Garay, S. Kutten, and D. Peleg, "A sub-linear time distributed algorithm for minimum-weight spanning trees," in *SIAM J. Comput*, 1998, pp. 659–668.

[17] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proc. IEEE INFOCOM '00*, 2000, pp. 585–594.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.