A Network Coding Approach to Reliable Broadcast in Wireless Mesh Networks*

Zhenyu Yang, Ming Li, and Wenjing Lou

Department of Electrical and Computer Engineering,
Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609
{zyyang,mingli,wjlou}@wpi.edu

Abstract. Reliable broadcast is an important primitive in wireless mesh networks (WMNs) for applications such as software upgrade, video downloading, etc. However, due to the lossy nature of wireless link, it is not trivial to achieve the reliability and efficiency at the same time. In this paper, we 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 based on ETX metric. The broadcast overhead and delay are simultaneously reduced by enabling each node to be covered by the parent node in the tree which promise its reliable reception of the whole file. Opportunistic overhearing is utilized to further reduce the number of transmissions. Extensive simulation results show that R-Code always achieves 100% packet delivery ratio (PDR), while introducing less broadcast overhead and much shorter delay than AdapCode.

1 Introduction

Wireless mesh networks (WMNs) is an approach to provide high-bandwidth network access for a specific area, which becomes prosperous during the last decade. 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. 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 byte 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 means consuming minimal amount of network bandwidth and complete the broadcast process quickly.

The fundamental challenge in the design of reliable and efficient broadcast protocol in WMNs is the unreliability of wireless links, which is mainly due to the path loss, interference and channel fading [12]. Previous schemes usually achieve reliability by applying the same mechanisms as used in wired broadcast protocols, such as ARQ mechanism [13], FEC mechanism [11], etc. However, these schemes tend to introduce large amount of redundant transmissions and incur problems like "ACK explosion", etc.

* This work was supported in part by the US National Science Foundation under grants CNS-0626601, CNS-0746977, CNS-0716306, and CNS-0831628.

B. Liu et al. (Eds.): WASA 2009, LNCS 5682, pp. 234–243, 2009.

Network coding (NC) has emerged as a promising technique to increase the network bandwidth-efficiency and reliability recent years. Briefly speaking, NC is a new communication paradigm for packet-based networks which breaks with the conventional store-and-forward way. It gives the intermediate nodes the flexibility of encoding different packets received previously together for subsequent transmission with the purpose of benefiting multiple receivers with single transmission. Those packets mixed together could be from different data flows, in which case it is called inter-flow nework coding [8]; otherwise if they are from the same data flow, it is called intra-flow network coding [2]. Moreover, NC makes the schedule of retransmission easier since now each encoded packet is the same and no specific packet is indispensable for the receivers.

Prior works that exploit the advantage of NC for reliable broadcasting in wireless networks are still at preliminary stage. As far as we know, AdapCode [7] is the only protocol designed purposely for reliable broadcasting. It combines probabilistic forwarding and NC together and promises perfect reliability by a "NACK+timer" mechanism.

In this paper, we propose R-Code, a NC-based reliable broadcast protocol in WMNs with unreliable links. In R-Code, we build a minimum spanning tree (MST) to behave as a "virtual backbone", whose link weight is based on ETX metric [4]. *The key idea of R-Code is that every node is covered by the best neighbor, the parent node in the MST, for its reliable reception and successful decoding.* R-Code continues to exploit the broadcast nature of wireless transmission to further reduce the transmissions. Simulation results show that, R-Code can guarantee 100% PDR with less broadcast overhead than AdapCode and achieve much lower broadcast delay at the same time.

The rest of the paper is organized as follows. Related work in given in Section 2. We describe the network model and network coding primitives in Section 3. The design of the protocol is shown in Section 4. In Section 5, we present the simulation results. In the end, we give conclusion in Section 6.

2 Related Work

Exploiting the idea of NC for reliable broadcast is still at a preliminary stage. MORE [2] is the first NC-based protocol for reliable routing. To the best of our knowledge, it is also the only one that is implemented and applied in real world scenario. MORE combines the idea of opportunistic routing (OR) [1] and network coding, which eliminates OR's requirement for complicated and costly coordination between receivers while enjoying the throughput benefit of NC. MORE is designed for supporting unicast flows and it can also be extended to support broadcast applications, although it does not perform very well in that case, because almost all the nodes become forwarders which incur heavy contention and congestion. Koutsonikolas *et.al.* proposed Pacifier [10], a high throughput, reliable multicast protocol based on MORE. Pacifier addresses the weakness of MORE by maintaining a multicast tree structure. It also alleviates the "crying baby" problem by a round-robin algorithm and further increases the throughput. 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. We note that although both MORE and Pacifier can support broadcast applications, they are not purposely designed for that. In both protocols, the source node is

the only active "pump" that transmits packet proactively and all the other intermediate nodes just play the role of forwarder, which passively relay what they received. Thus, for some missing packet, the destinations can only get it from the source rather than some node nearby that has already got the whole file, which will incur many redundant transmissions and is inefficient.

In comparison, since each node has to get the whole file ultimately, the schemes specially designed for broadcast make all the nodes within the network to be "temporary source" after receiving the whole file. AdapCode [7] is such a protocol, which aims at reliable broadcast in wireless sensor networks (WSNs) and studies the global code updates. It also tries to achieve load balance and rapid propagation. Since we will compare R-Code with AdapCode in this paper, a brief overview of it is presented below. AdapCode is motivated by probabilistic forwarding approach, which means for each received packet, the receiver just forwards it with some probability less than 1 in order to reduce the traffic introduced compared with naive flooding. AdapCode combines this idea with network coding which further reduces the traffic by letting a node send one coded packet after receiving every *N* coded packets from other nodes. The particular choice of such *N* is called *coding scheme*. Also, it applies a "NACK+Timer" mechanism to promise 100% reliability. During the code update process, each node keeps a count-down timer. If the node missed some packets, it will broadcast a NACK packet when the timer fires, within which contains the IDs of those missed packets. All the other nodes overheard this NACK and have already received those requested packets will take part in a response process and one of them will be randomly selected as the real responder. This process goes on until all the nodes get all the packets. However, since the NACK mechanism inherently tends to elongate the reception time, 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.

3 Preliminaries

3.1 Network Model

In this paper, We only consider the wireless mesh network that consists of all the routers, one of which plays the role of gateway that connects to the Internet. The gateway is always the only source that wants to broadcast files through the network. Since we only consider the one-to-all scenario, intra-flow network coding [2] is adopted to reduce the number of transmissions and simplify the protocol design. The WMN is modelled as a weighted undirected graph $G(V, E)$, where V is the set of nodes 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 (the reason for taking the average of probabilities of both directional is that the construction of MST is based on undirected links). Since in reality mesh routers are often statically deployed, we assume the network topology and also the link quality keep stable during one broadcast session [14], which is usually finished within several minutes at most. Moreover, we assume the routers have enough memory that can store several generations

simultaneously, each of which usually are tens of kilo-bytes. The definition of generation is presented in the following section.

3.2 Network Coding

For the purpose of reducing the complexity of encoding/decoding and storage requirement, the broadcasted file is divided into segments sequentially, called generation [3]. Each generation contains same number of k packets, denoted as $s_i, i = 1, 2, ..., k$. All the encoding/decoding operation is done within one generation, where a coded packet x is the linear combination of all these k original packets: $x = \sum_{j=1}^{k} \alpha_j s_j$. $< \alpha_1, \alpha_1, ..., \alpha_k >$ is the encoding vector, which identifies how to generate this coded packet from the original packets in this generation. Each element of the coding vector is independently randomly [6] selected from a Galois field $GF(2^q)$. Every coded packet includes this coding vector in the packet header for future decoding. Upon receiving a coded packet x of generation i, the receiver puts x into the buffer matrix for generation i and then tries to decode it along with all the other coded packets of this generation received previously by doing Gaussian elimination on the buffer matrix, whose complexity is $O(k^3)$.

3.3 Minimum Spanning Tree

Since applying MST for broadcast is well studied in both wired and wireless networks, many efficient and distributed algorithms of building and maintaining MST for a given network are proposed [5, 9],. 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 and be shared by multiple broadcast sessions, thus the extra communication and computation overheads introduced can be amortized and are negligible [2].

4 R-CODE

In this section, we present the design of R-Code. At first, we use a simple example to explain the intuition behind R-Code; In the following, we present the details of the design.

4.1 Intuition of R-Code

The intuition underlying our approach is that for each node i , since all its neighbors capable of behaving as a "temporary source", it can always choose to be reliably covered by the best neighbor with minimum cost, where the word best means this neighbor can transmit one packet to i reliably with minimum expected number of transmissions. We believe a good global performance can be achieved through all those simple, optimal local decisions.

This can be illustrated with a simple example that is shown by Fig.1. This toy network consists of 4 nodes, with S being the only source and wants to broadcast a packet

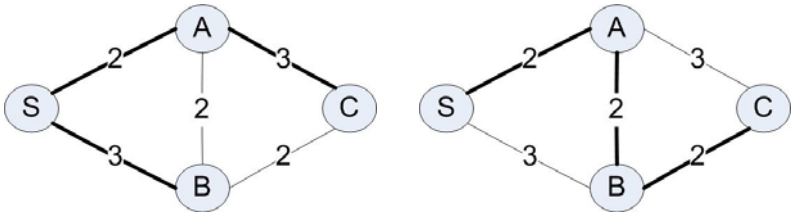

Fig. 1. A simple example to show the intuition of R-Code. The ETX value is assigned for each link.

reliably. If we build the broadcast tree like Pacifier does, which is to combine all the best unicast path from source to every other nodes, then we get the tree that is shown in Fig.1(a) with bold lines. The total expected number of transmissions introduced is 6, where both of S and B generates 3 transmissions; However, we observe that some nodes are not covered by the best choice. For example, C can get the packet from B with 2 transmissions, which is more efficient than getting it directly from the source S. If we make all nodes to be covered by their best neighbors, then we naturally build a MST as broadcast tree, which is shown in Fig.1(b). Now the total expected number of transmissions needed is $2 + 2/3 + 14/9 = 4.23$, which is generated by S, B, C respectively. Actually, this is also the minimum number of transmissions needed. However, we do not claim that this optimal result can always be achieved through R-Code. We note that our effort in this paper is put on designing practical and efficient protocol rather than pursing theoretically optimal performance.

4.2 Design of R-Code

Generally, R-code can be divided into two stages.

Initialization Stage. During this stage, each node i broadcasts "Hello" packets with period T to estimate the quality of links to other nodes nearby. Based on collected information, 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. Further, based on this table, i builds a MST [9] and stores this tree structure by a node set which contains all the neighbors within the MST.

Above is the general initialization stage. For a specific broadcast session, the single source s makes the MST just built a directed tree originated from itself, which is the root. 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.

Broadcast Stage. R-Code works on top of the IP layer and the packet header format is shown in Fig.2, which contains a type field that identifies data packet from ACK packet, the source's IP address, broadcast session id, generation index, generation size which

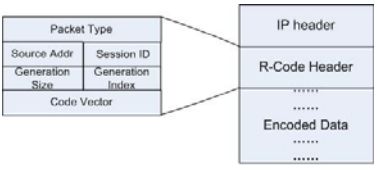

Fig. 2. R-Code packet header format

indicates the total number of generations and code vector that describes the packet content with respect to the original packets. The broadcast session starts by letting the source continuously send coded packets of the first generation with interval T_{pkt}. When a node i overhears a packet, it stores this packet in the corresponding buffer matrix and runs Gaussian elimination on the matrix to check if it received enough information to retrieve all the original packets of this generation. If so, it records this reception time and notifies the guardian by sending a positive ACK in unicast, whose reliability can be promised by MAC layer. Then, if i is currently not in the process of broadcasting for other generation, it begins to play the role of guardian for its children by keeping sending coded packets of this generation with interval T_{pkt}. Otherwise, if i is still not able to retrieve all the original packets of this generation, it keeps silent and waits for more packets;

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 the corresponding buffer matrix. Additionally, it should add the index of this generation into the successfully acknowledged ones and pause for a period of $T_{generation}$ time before broadcasting the next generation, allowing the children nodes to rebroadcast what they just received [15]. There are two challenges need to be solved to make R-Code run smoothly.

1. How to select the generation to be sent next? For source node, it 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 termination of the broadcast session, which maybe still goes on in other nodes. For other transmitters, the selection of next generation follows FIFO policy: it chooses the earliest received generation from those successfully received but not successfully acknowledged ones, the index of which 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 generations have been acknowledged. If so, it quits this broadcast session; or else this node keeps silent and wait for future reception.

2. Dealing with ACKs. As presented above, the perfect reliability is guaranteed by the guardian-child relationship. Although multiple ACKs will be send back to the guardian, we argue that this will not incur "ACK explosion" problem for reasons listed below. (1) 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, for some specific guardian node, the link qualities between it to its children are different and

Table 1. Optimal coding schemes

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

Table 2. Simulation parameters

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

those child nodes tend to receive the whole generation successfully at different times with very high probability. (2) Since the child nodes are usually a small subset of all the neighbors of the guardian node, thus even some of them really reply ACKs almost at the same time and cause a conflict at the guardian node side, the collision avoidance mechanism of 802.11 can easily handle this. Further, we require that each child backoffs a random short time before sending ACK.

5 Performance Evaluation

We evaluate the performance of R-Code and compare it with AdapCode through extensive simulations. In our implementation of AdapCode, for fairness, we always allow the nodes to generate coded packets by doing linear combination of all the k original packets rather than a portion of them, where the latter is AdapCode's intended consideration for WSNs. As claimed in [7], this relaxation could give adapCode better performance on bandwidth efficiency.

5.1 Simulation Settings

We use Glomosim simulator in our simulations. The network consists of a 7×7 grid of static nodes, where the grid size equals $200m$ and average number of neighbors per node is 9.51. For AdapCode, We follow the optimal coding schemes presented in [7], which is shown in Table 1.

The source is fixed to be node 0 in the left-bottom corner for all the simulations. The broadcasted file is 1MB and is divided into 4096 pieces, each of which is 256 byte long. The MAC layer runs $802.11b$ with some modification that fixes the data rate to be $11Mbps$. Other related simulation parameters are listed in Table 2. We run both protocols in this network 10 times with T_{pkt} varied from $11ms$ to $29ms$ and use the following metrics for comparison:

Average propagation delay: The total time required for a node to receive the whole file, average 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 linearly dependent packets: The total number of linearly dependent packets received by all the nodes divided by the node number. We note that this number includes those packets overheard by some node who has already got the whole generation that this packet belongs to.

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

5.2 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 guard relation and ACKs of R-Code. From Fig.3(a) we can see that R-Code introduces fewer transmissions than AdapCode in all settings and the performance gain is greater when the per packet broadcast interval T_{pkt} is larger, where the maximal gain can be 15% when T_{pkt} is $29ms$. We also observe that as T_{pkt} increases, the average number of transmissions incurred by R-Code decreases while the average traffic introduced by AdapCode keeps almost the same and even increases a little. The key reason for this is R-Code's local optimal decision. For each node, it always choosing the best neighbor to be the guardian. In comparison, when a node i in AdapCode is requiring 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. The second cause for the more number of transmissions of AdapCode is presented below. 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, its timer will fire firstly and it will broadcast the NACK. Unfortunately, this is just what happened to those nodes with bad connection to the sender. Further, because this bad connected node has received much fewer packets, it will require more retransmissions in its NACK and most of those retransmissions tend to be useless for those good connected receivers who only miss small number of innovative packets. This is shown clearly in Fig.3(b), where we can see that nodes in AdapCode encounter more linearly dependent reception in all cases.

5.3 Propagation Delay

Compared with the performance of introduced traffic, R-Code brings a larger gain over AdapCode when it comes to the metric of propagation delay. It is obviously to see that under all settings, the average propagation delay of R-Code performances much better than AdapCode, which is shown in Fig.4(a). The gain is higher when packet broadcast interval is small, i.e., when T_{pkt} is $11ms$, the reduction ratio can be almost 50%. This is consistent with our analysis in previous section, which indicates that NACK mechanism inherently tends to elongate the propagation delay. And we also observe that both protocols' broadcast delay grow almost linearly to the packet broadcast interval. This is

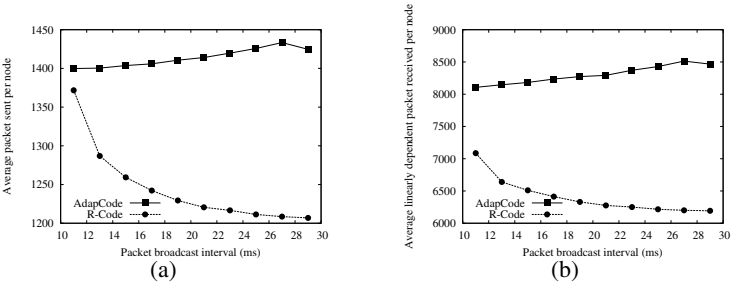

Fig. 3. Traffic

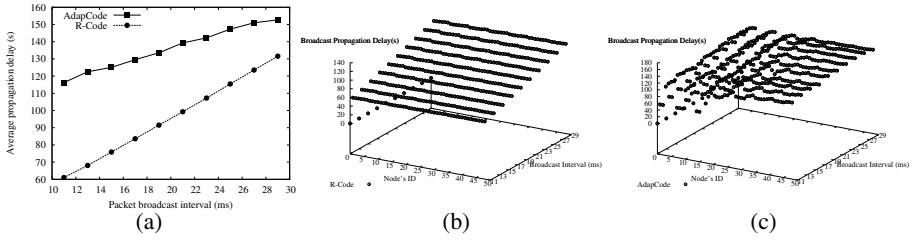

Fig. 4. Propagation Delay

because all the transmitters inject packets into network with a period of T_{pkt}. Besides shorter average propagation delay, R-Code can also achieves more consistent individual node propagation delay, which means all the nodes running R-Code can get the whole file at approximately the same time. This is shown in Fig.4(b); In comparison, nodes running AdapCode receive the whole file with delays that have much larger variation, which is shown in Fig.4(c). This makes R-Code more appropriate for applications like software updates where some subsequent operations will be done just after the reception of the whole file.

From the simulation results presented above, we can see that R-Code is a simple and high performance reliable broadcast protocol for WMNs, which achieves less transmission overhead and shorter broadcast delay without the need of complicated timer mechanism. However, we also observe that there is a tradeoff between those two metrics, both for R-Code and AdapCode. Thus for specific application, it should chooses proper parameter values according to its own requirement.

6 Conclusion

In this paper, we focus on designing a practical broadcasting protocol for wireless mesh networks, which provides 100% reliability for all receivers. We present R-Code, A simple, efficient and high-performance broadcast protocol with the help of network coding to reduce the number of total transmissions required and average propagation delay. The core idea is to promise each node to be covered by the best neighbor. Based on

the intuition that local optimal solution can achieves better global performance, we apply MST as the broadcast tree. Extensive simulations showed that R-Code can reduce the number of required transmissions and propagation delay as high as 15% and 50%, respectively, compared with the state-of-the-art AdapCode.

References

1. Biswas, S., Morris, R.: Opportunistic routing in multi-hop wireless networks. SIGCOMM Computer Communications Review (2004)
2. Chachulski, S., Jennings, M., Katti, S., Katabi, D.: Trading structure for randomness in wireless opportunistic routing. In: SIGCOMM 2007 (2007)
3. Chou, P., Wu, Y., Jain, K.: Practical network coding. In: Proceedings of the 41st Allerton Conference on Communication, Control, and Computing (September 2003)
4. De Couto, D., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multihop wireless routing
5. Garay, J.A., Kutten, S., Peleg, D.: A sub-linear time distributed algorithm for minimum-weight spanning trees. SIAM J. Comput. (1998)
6. Ho, T., Mdard, M., Koetter, R., Karger, D.R., Effros, M., Shi, J., Leong, B.: A random linear network coding approach to multicast. IEEE Trans. Inform. Theory (2006)
7. Hou, I.-H., Tsai, Y.-E., Abdelzaher, T., Gupta, I.: Adapcode: Adaptive network coding for code updates in wireless sensor networks. In: INFOCOM 2008 (April 2008)
8. Katti, S., Rahul, H., Hu, W., Katabi, D., Medard, M., Crowcroft, J.: Xors in the air: practical wireless network coding. SIGCOMM Computer Communications Review (2006)
9. Khan, Maleq, Pandurangan, Gopal: A fast distributed approximation algorithm for minimum spanning trees. Distributed Computing (April 2008)
10. Koutsonikolas, D., Hu, Y.-C., Wang, C.-C.: High-throughput, reliable multicast without crying babies in wireless mesh networks. In: INFOCOM 2009 (April 2009)
11. Koutsonikolas, D., Hu, Y.C.: The case for fec-based reliable multicast in wireless mesh networks. In: DSN 2007: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 491–501 (2007)
12. Marco Zú, N.Z., Krishnamachari, B.: An analysis of unreliability and asymmetry in low-power wireless links. ACM Trans. Sen. Netw. 3(2), 7 (2007)
13. Pagani, E., Rossi, G.P.: Reliable broadcast in mobile multihop packet networks. In: MobiCom 1997, pp. 34–42 (1997)
14. Reis, C., Mahajan, R., Rodrig, M., Wetherall, D., Zahorjan, J.: Measurement-based models of delivery and interference in static wireless networks. SIGCOMM Comput. Commun. Rev. 36(4), 51–62 (2006)
15. Scheuermann, B., Lochert, C., Mauve, M.: Implicit hop-by-hop congestion control in wireless multihop networks. Ad Hoc Networks (2008)