Network Coding Based Reliable Broadcast Protocol in Multi-Channel Multi-Radio Wireless Mesh Networks

Xiaobin Tan, Hong Wen, Kangqi Wang

Department of Automation, University of Science and Technology of China, Hefei, China, 230027
Email: xbtan@ustc.edu.cn, dreamone@mail.ustc.edu.cn, wannkq@mail.ustc.edu.cn

***Abstract*—Multi-Channel Multi-Radio (MCMR) Wireless Mesh Networks (WMNs) have emerged as a new paradigm in multi-hop wireless networks. In a typical MCMR WMNs, each node has multiple radios with multiple available channels on each radio, which allows nodes to have simultaneous transmissions and receptions. Therefore, network performance is improved. As a key technology in WMNs, reliable broadcast can provide efficient data transmission. GreedyCode is a network coding based reliable broadcast protocol proposed by our group earlier, whose basic idea is to opportunistically select the forwarders with the highest transmission efficiency to transmit the encoded packets while the neighbors just listen. In this paper, we consider one-to-all broadcast scenarios and propose a novel GreedyCode based reliable broadcast protocol MCMR-GreedyCode, which is two-fold: channel assignment and link scheduling. Specially, we propose the Level Channel Assignment Strategy (LCAS) algorithm and determine the number of data packets to be sent each time according to the feedback information from one-hop neighbor nodes. In addition, any intermediate node that receives complete data can forward data to those nodes that don't. The process repeats until all destination nodes receive complete data. Simulation results show that MCMR-GreedyCode has lower network latency and greater throughput than some existing network protocols, such as GreedyCode, MCM, MLRM, etc.**

I. Introduction

Multi-Channel Multi-Radio (MCMR) Wireless Mesh Networks (WMNs) have emerged as a new paradigm in multi-hop wireless networks. They enhance network-wide throughput by parallelizing packet forwarding on multiple channels and radios. They also can be used on the "last mile" problem for extending or enhancing Internet connectivity [1].

As a key technology in WMNs, reliable broadcast can provide efficient data transmission, a fundamental problem for which is how to make sure each node receive complete message from the source node correctly despite the low-quality link.

Proposed by Ahlswede et al. [2], Network Coding(NC) will bring a performance boost to the network. By mixing the contents of multiple packets before forwarding, and broadcasting different encoded packets through intermediate nodes, NC can reduce duplicate transmission apparently. And categorizing NC into intra-flow NC and inter-flow NC motivates us to apply it to WMNs.

MORE [3], a MAC-independent opportunistic routing, is the first multicast routing protocol based on intra-flow NC. MORE randomly mixes packets before forwarding, which ensures that routers which hear the same transmission do not forward the same packets. Therefore, MORE doesn't need special scheduler to coordinate routers and can run directly on 802.11. However, MORE has the "crying baby" problem, which seriously degrades the performance. Pacifier [4], a new high-throughput reliable multicast protocol for WMNs, seamlessly integrates four building blocks: tree-based opportunistic routing, intra-flow network coding, source rate limiting, and round-robin batching. It supports high-throughput, reliable multicast routing in WMNs, and solves the "crying baby" problem in MORE effectively. R-Code [5], a reliable and working broadcast protocol based on intra-flow NC, constructs a Minimum Spanning Tree (MST) as the backbone, where the weight of each link equals the ETX (Expected Transmission Count) on that link. GreedyCode [6], a greedy strategy based reliable broadcast protocol proposed by us earlier, abandons the multicast tree, trying to make full use of all links, and opportunistically actives the forwarding nodes with the highest transmission efficiency within one-hop area to broadcast the encoded packets.

However, these protocols are all based on single channel single radio (SCSR) network environment. To further improve network performance, MCMR technology in WMNs is proposed: there exists several orthogonal channels with each node equipped with several interfaces. Besides, assigning a channel to one link is necessary for one link to exist. Therefore, in order to attain a higher performance in MCMR WMNs, an excellent channel assignment strategy is needed. M.Jahanshahi et al. [7] proposes a fundamental design issue for joint multicast routing and channel assignment in MCMR WMNs and provides a mathematical frameworks called binary integer programming(BIP) model. However, it is worth mentioning that the framework is centralized and the computation cost is very high, especially in large-scale networks. What's more, it doesn't take link quality into account. Zeng [8] proposes MCM algorithm, which first builds a multicast structure by minimizing the number of relay nodes and hop count distances between the source and destinations, then uses dedicated chan-

978-1-4673-5939-9/13/$31.00 ©2013 IEEE

nel assignment strategy to improve the network performance. But it ignores some potential links as well as the influence of link quality. Then, Zeng proposes another multicast algorithm MLRM [9] that considers link quality. However, it still ignores some potential links by constructing a multicast tree structure as its backbone. Although MLRM is a great improvement over MCM, it only considers a tree as its backbone. In this paper, we propose to effectively use all the links, it is no doubt that the network performance will be better.

In this paper, we put forward a novel GreedyCode based reliable broadcast protocol MCMR-GreedyCode, which is two-fold: channel assignment and link scheduling. Specially, we propose the Level Channel Assignment Strategy (LCAS) algorithm and determine the number of data packets to be sent each time according to the feedback from one-hop neighbor nodes in MCMR-GreedyCode. Besides, when one node is about to send data, it makes channel assignment first, then link scheduling. The process repeats until all destinations receive complete data.

The rest of this paper is organized as follows: In Section II, we introduce the system model and the basic idea of GreedyCode. In Section III, we describe MCMR-GreedyCode in detail. In Section IV, we evaluate the performance of MCMR-GreedyCode. Finally, we draw the concluding remarks in Section V.

II. PRELIMINARIES

In this section, we will introduce the underlying network model for MCMR WMNs as well as the basic idea of GreedyCode protocol proposed by our group earlier.

A. System Model

To simplify the system model, the network is considered as a graph $G = (V, E)$, where V is the set of all nodes and E is the set of all links among neighboring nodes. Two nodes u and v are directly connected and form a communication link (u, v) if they are within the transmission range of each other and share a common channel. In addition, each link has one weight $p(i, j)$ defined as the transmission probability from node i to node j, and it holds that $p(i, j) = p(j, i)$. In our system model, there exists only one source node which has a volume of data to broadcast to all destination nodes. Those nodes that receive complete data will play the role of forwarding nodes meaning that they can also transmit data. Besides, when two nodes that share a link in common both have received complete data, the link can be deleted due to its uselessness for data transmission.

B. The Idea of GreedyCode

GreedyCode [6] abandons the structure of the multicast tree, trying to make full use of all links, and dynamically selects the forwarders having the largest OBT (One-hop Broadcast Throughput) within one-hop area to forward data. OBT is defined as follows:

$$OBT(i) = \begin{cases} \sum_{j \in \mathcal{U}_i} \frac{1}{w_{i,j}} & \text{if node } i \text{ has received the whole batch;} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Where, $u(i)$ is the neighbor set of forwarding node i, which has not received the whole batch. $w_{i,j}$, defined as the reciprocal of $p_{i,j}$, is the ETX of link (i, j).

Next, we will illustrate the efficiency of GreedyCode by a simple example. Considering a network consisted of five nodes and seven links, where S is the source node which has a batch of k packets to deliver to all the other target nodes A, B, C, D. Fig.1 shows the transmission process of GreedyCode.

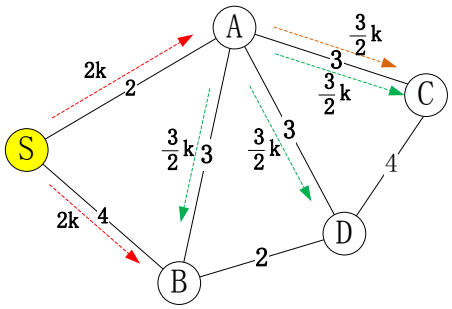

Fig. 1: **Simple example of GreedyCode.**The number on each link represents the ETX of that link, that is, the reciprocal of transmission probability. First of all, source node S broadcasts 2k packets, and then A broadcasts $\frac{3}{2}k$ packets, further A broadcasts $\frac{3}{2}k$ packets again, so it will take 5k packets' transmission time to guarantee all destination nodes receive complete data.

Firstly, source node S broadcasts $2k$ packets, guaranteeing the reception of k packets at node A and $\frac{1}{2}k$ packets at node B on average, respectively. Then node A transmits $\frac{3}{2}k$ packets, making node B, C, D receive k, $\frac{1}{2}k$, $\frac{1}{2}k$ packets on average respectively. Furthermore, both node C and D will receive k packets on average after $\frac{3}{2}k$ packets' transmission at node A. Therefore, it takes $2k + \frac{3}{2}k + \frac{3}{2}k = 5k$ packets' transmission time to guarantee all the target nodes receive k packets, which has a better performance than Pacifier and R-Code [6].

In fact, the performance of GreedyCode is limited due to SCSR technology, where each node can only send or receive message each time. However, nodes can have simultaneous transmissions and receptions if MCMR technology is used, which motivates the emergence of our protocol: MCMR-GreedyCode.

III. MCMR-GREEDYCODE

In GreedyCode [6], only single channel, single radio is considered, which limits further improvement of network performance. Therefore, we propose a novel MCMR based protocol MCMR-GreedyCode, which is two-fold: channel assignment and link scheduling. Next, we'll describe them in detail.

A. Channel Assignment Strategies

Assigning a channel to one link is a necessary condition for data transmission on it. But the problem is how to make the proper strategy for channel assignment before transmitting data. With respect to the problem, We put forward a novel channel assignment strategy: Level Channel Assignment Strategy (LCAS), whose basic idea is to assign channels to links by several rounds until all the links have been assigned channels

or the number of assigned channels equals to the radio number of nodes. At each round, the intention of LCAS is dynamically selecting a channel to assign for the purpose of reducing the existed interferences among links to the least. And the detail process of LCAS is described as Algorithm 1.

Algorithm 1 The process of LCAS algorithm

Require: S: any node that tends to send data.
Set_Neigh: the neighbor set of node S, which have not received data fully.
UCS: the set of unassigned channels.
C: the total number of orthogonal channels.
$Link\_Set(c)$: if assigning channel c, no interference links existed between S and Set_Neigh.
$Find\_Interference(cc, S, Set\_Neigh)$: if assigning channel cc, the total interference among links.
$Find\_Link\_Set(t, S, Set\_Neigh)$: finding no interference links existed between S and Set_Neigh if assigning channel t.
Ensure: $UCS = \{1, 2, 3, .., C\}$
while $Set\_Neigh \neq \{\emptyset\}$ and $UCS \neq \{\emptyset\}$ **do**
  $k \Leftarrow +\infty$
  for all cc in UCS **do**
    $y \Leftarrow Find\_Interference(cc, S, Set\_Neigh)$
    if $y < k$ **then**
      $t = cc$
      $k = y$
      $Link\_Set(t) = Find\_Link\_Set(t, S, Set\_Neigh)$
    end if
  end for
  for all $Link(S, n)$ in $Link\_Set(t)$ **do**
    Assigning Channel t to link(S, n)
    Excluding n From Set_Neigh
  end for
  Excluding t From UCS
end while

Next, we illustrate the principles of LCAS algorithm using a simple example. Given two orthogonal channels 1, 2 with each node equipped with 2 radios. What we'll do is to assign channels to the links between node S and its neighbor nodes A, B, C, D. Besides, channel 1, 2, 1 has been assigned to link(A, E), (B, F), (C, G) respectively (Fig.2).

In the first round, node S can select an unsigned channel from $\{1, 2\}$. If selecting channel 1, link (S, A) will interfere with link(A, E), so will link (S, C) and (C, G). However if selecting channel 2, only one pair of links, namely link(S, B) and (B, F), will interfere with each other. Therefore, to make the interferences least, it is wise to assign channel 2 to link(S, A), (S, C) and (S, D). In the second round, assigning channel 1 to the link(S, B) directly is ok. In a nutshell, the channel assignment for Fig.2 is finished according to the LCAS algorithm.

After assigning channels, the next problem to be solved is the link scheduling each time.

B. Link Scheduling

Link Scheduling is to mainly deal with the problem that how many data packets to be sent each time according to the feedback information from one-hop neighbor nodes. In our view, any forwarding node, saying i, will stop the data transmission this time only when at least one neighbor node receives complete data. But the question is how many packets to be sent for node i before stopping.

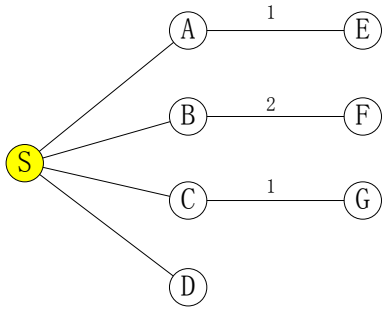

Fig. 2: **Simple example of LCAS algorithm.** The numbers on Link(A, E), Link(B, F), Link(C, G) represent assigned channel labels.

Given set_neigh_I represents these nodes that have not received complete data packets within one-hop area of node I.

For $\forall j \in set\_neigh_I$, as our assumption, node j not only locates within one hop local area of node I, but also within one hop local area of other nodes that have received complete data, the set of which is defined as set_other_{Ij}.

Therefore, in order to make node j receive complete data packets, it faces a problem that how many packets to be sent simultaneously for node i and nodes of set_other_{Ij}. This can be formatted as the following expression:

$$Out_j = \frac{|T| - hr_j}{p_{I,j} + \sum_{n \in set\_other_{Ij}} p_{n,j}} \tag{2}$$

where, $|T|$ is the number of complete data packets that node j is tend to receive, hr_j is the received number of data packets at node j, $p_{I,j}$ is the transmission probability from node I to node j.

After node j receives complete data packets, it will send feedback message to node I as well as each node of set_other_{Ij} by a common channel. Thus, node I updates the number of data packets to be sent this time from the feedback message, which is defined as follows:

$$Tr\_Data_I = Min_{j \in set\_neigh_I}\{Out_j, Tr\_Data_I\} \tag{3}$$

So does each node of set_other_{Ij}. What's more, if the number of nodes that can send data to node j exceeds the total number of its interfaces k, then node j will select those nodes with top k OBT to receive data.

C. A simple example of MCMR-GreedyCode

Fig.3 shows a topology the same as Fig.1. Given existing 3 orthogonal channels and 2 interfaces at each node.

First of all (Fig.3(a)), source node S broadcasts $2k$ packets after assigning channel $c = 1$ to link(S, A) and link(S, B), which will lead the reception of k packets at node A and $\frac{1}{2}k$ packets at node B, respectively.

In addition (Fig.3(b)), after the reception of complete data at node A, the link(S, A) can be deleted. Furthermore, channel $c = 2$ is assigned to link(A, B), (A, C) and (A, D). After this, node A and node S will broadcast $\frac{6}{7}k$ data packets simultaneously to guarantee the complete reception of k data

packets at node B. Meanwhile, $\frac{2}{7}k$ data packets are received by node C, node D respectively.

What's more (Fig.3(c)), the links (B, S), (B, A) can be deleted after the reception of complete data at node B. Then node A and node B forward $\frac{6}{7}k$ data packets simultaneously to guarantee the reception of k packets reception at node D after assigning channel $c = 3$ to link(B, D).

Finally (Fig.3(d)), after assigning $c = 1$ to the link(D, C), node A and node D forward $\frac{36}{49}k$ data packets simultaneously to node C, which guarantees its complete reception of data.

In a nutshell, to guarantee the complete reception of all the destination nodes, the transmission time of $2k$+$\frac{6}{7}k$+$\frac{6}{7}k$+$\frac{36}{49}k$ (about $4.45k$) packets is necessary. Therefore, the performance of MCMR-GreedyCode is superior to those of Pacifier, R-Code and GreedyCode. In fact, node S can forward data to node A and B using radio 1 and 2 simultaneously, so do other nodes. Here, we only consider radio 1 for easy interpretation. Therefore, the real performance will be better, which means the total transmission time will be less than $4.45k$.

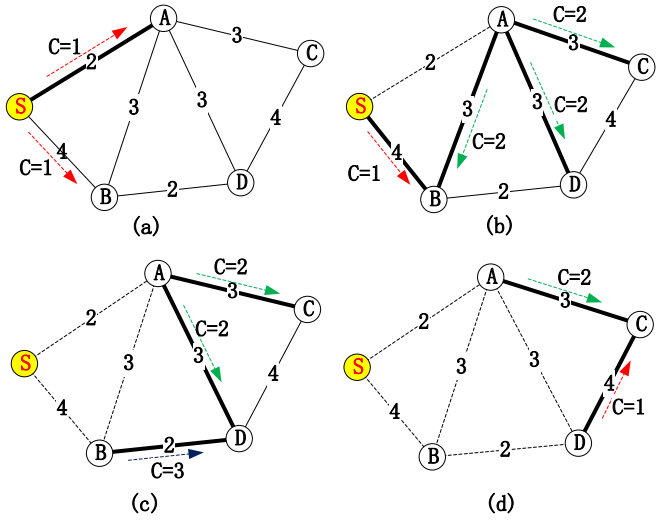

Fig. 3: **The transmission process of MCMR-GreedyCode.** The number on each link represents the ETX of that link, that is, the reciprocal of transmission probability. (a)shows S's transmission.(b)shows node S and A's simultaneous transmission.(c)shows A and B's simultaneous transmission. (d)shows the simultaneous transmission of node A, D.

D. Design of MCMR-GreedyCode

At the beginning, there is only one node called the source node that tends to broadcast packets to all the destination nodes. First of all, after making proper channel assignment using LCAS algorithm for the links between the source node and its one-hop neighbors, the source node will keep broadcasting encoded packets of the current batch until at least one neighbor node receives the whole batch and decodes all the original packets. Then, a neighbor node, saying node i, broadcasts a notification message to its one hop neighbors by a proper channel. After this, the source node deletes the link between it and node i. Besides, any node that receive complete data can play the role of temporal source, in other words, it has the ability of transmitting data. Therefore, when one node is about to send data, it makes channel assignment firstly, then link scheduling. The similar process repeats until all destinations receive complete data.

IV. PERFORMANCE EVALUATION

In this section, the actual performance of MCMR-GreedyCode is investigated through simulation. The NS2 simulator program is used where network topology, physical layer and MAC layer are configured the same as those in [6], which are shown in Table I. And comparing our schemes with Pacifier, R-Code, GreedyCode as well as MCM, MLRM is our chief work. Since broadcast is a special case of multicast, MCM and MLRM can be easily adapted to be a broadcast algorithm. we concern the following metrics:

Average broadcast latency: The time it takes when the source just started to transmit data to the time when all destinations received the complete data.

Throughput: the total data volume that all the destination nodes receive in a unit time period.

TABLE I: Simulation parameters

Simulation parameters	Value
Radio propagation model	Two ray ground
Simulation area	400 m × 400 m
Number of nodes	16
Transmission range	160 m
Movement model	Random waypoint
Traffic type	CBR(UDP) and TCP
Antenna	Omni-Antenna
Bandwidth	10M
MAC type	Mac/802.11
Packet size	200

A. The Performance of MCMR-GreedyCode

We first investigate the performance of MCMR-GreedyCode with different numbers of channels and interfaces, and we set the batch size as 16. Fig.4 and Fig.5 show the results of the experiment.

From the results of Fig.4 and Fig.5, we can draw the conclusion that when the number of channels is fixed, the more the number of interfaces, the higher the throughput and the less the average broadcast latency. Besides, when the number of interfaces is fixed, the more the number of channels, the higher the throughput and the less the average broadcast latency.

Because there will be more numbers of links with the increasing numbers of channels and interfaces. Specifically, nodes can send or receive data from multiple interfaces simultaneously, which brings the performance boost.

B. Comparing with Pacifier,R-Code,GreedyCode

In this experiment, we set the number of orthogonal channels as 3, the number of interfaces as 2. Fig. 6 and Fig. 7 show the results of the experiment of Pacifier, R-Code, GreedyCode and MCMR-GreedyCode with a batch size of 8, 16, 32 and 64.

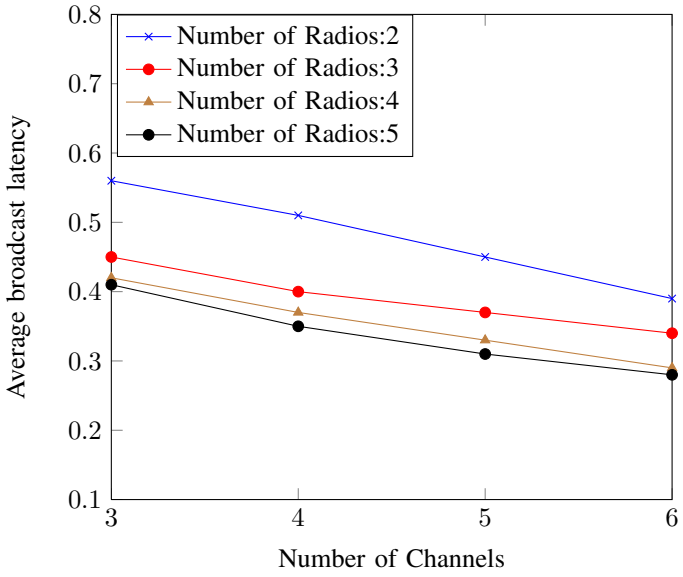

Fig. 4: Average broadcast latency for MCMR-GreedyCode with different numbers of channels and interfaces.

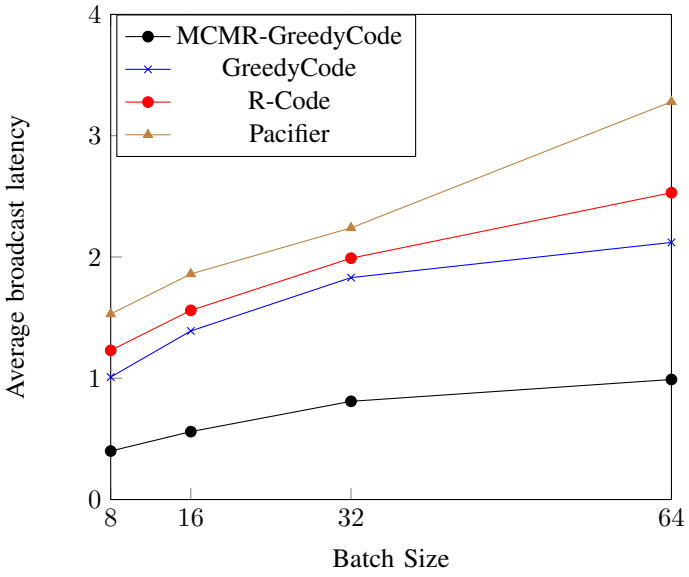

Fig. 6: In case of 3 orthogonal channel and 2 radios equipped in each node, average broadcast latency for Pacifier, R-Code, GreedyCode, MCMR-GreedyCode under different batch size.

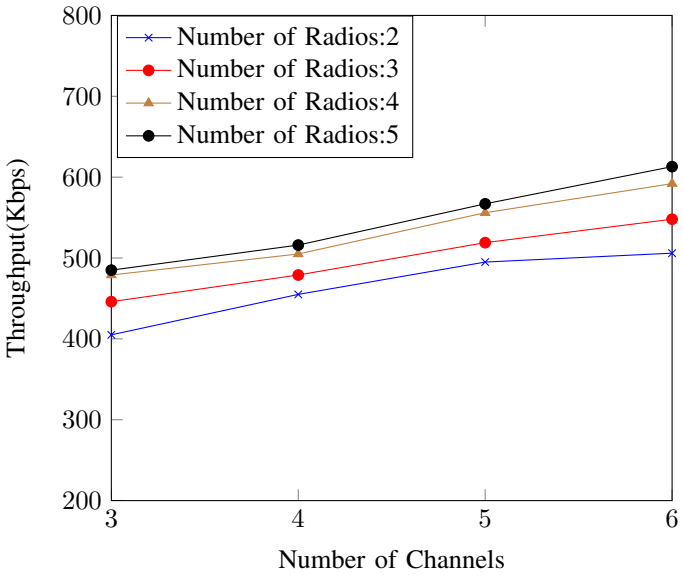

Fig. 5: Throughput for MCMR-GreedyCode with different numbers of channels and interfaces.

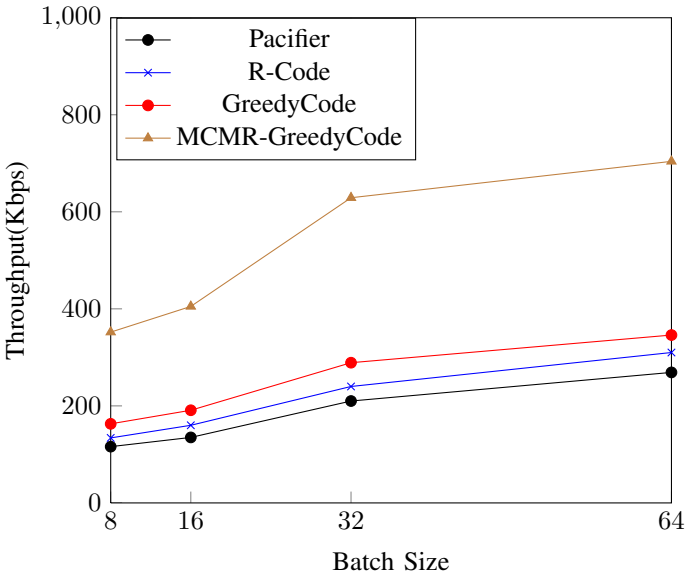

Fig. 7: In case of 3 orthogonal channel and 2 radios equipped in each node, throughput for Pacifier, R-Code, GreedyCode, MCMR-GreedyCode under different batch size.

We can conclude from Fig.6 and Fig.7 that average broadcast latency under different protocols will increase with the increasing of batch size, so will the throughput. Specially, MCMR-GreedyCode achieves the optimal performance among all the protocols due to the nature characteristic of multi-channel, multi-interface that each node can send or receive data from multiple interfaces simultaneously. Obviously, it increases the throughput and decreases the average broadcast latency.

C. Comparing with MCM and MLRM

According to the results in Fig. 4 and Fig. 5, the performance of MCMR-GreedyCode is improved to some extent when the number of interfaces is greater than 4. Therefore, we set the number of interfaces as 4 in the experiment. Fig.8 and Fig.9 show the results of the experiment.

It is obvious that the performance of MLRM which takes link quality into account is better than that of MCM. However, MLRM exploits the multicast tree only. As such, its performance is limited. MCMR-GreedyCode, on the other

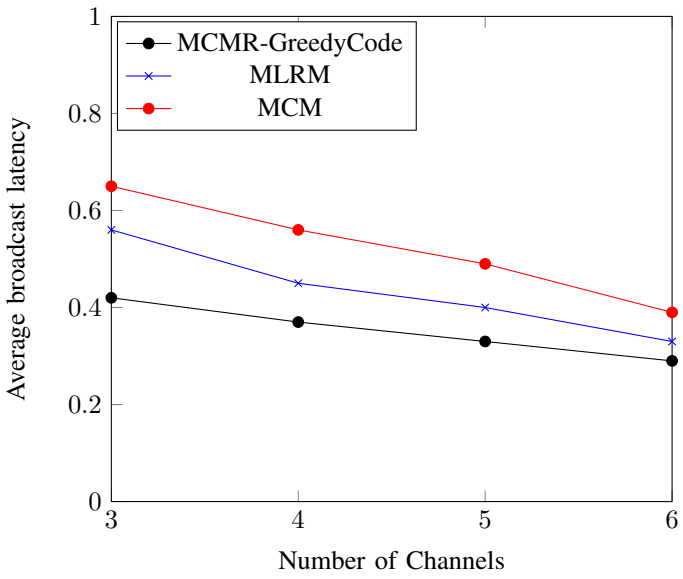

Fig. 8: Average broadcast latency for MCMR-GreedyCode, MLRM, MCM under 4 radios equipped in each node.

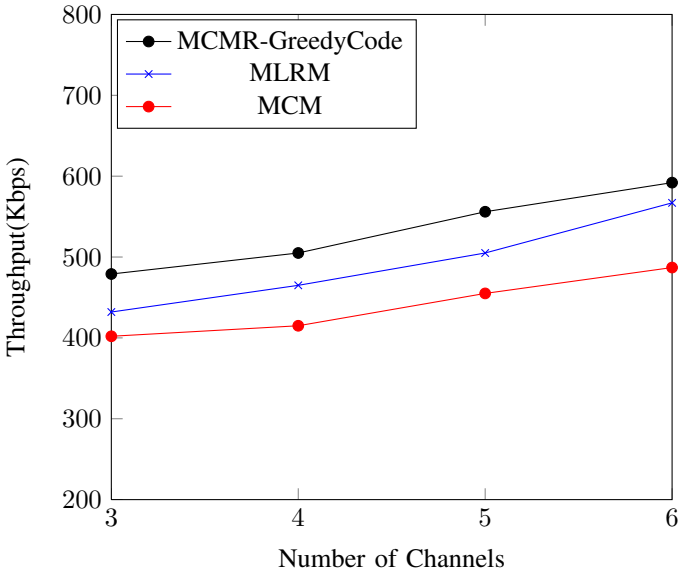

Fig. 9: Throughput for MCMR-GreedyCode, MLRM and MCM under 4 radios equipped in each node.

hand, makes use of all links, and is expected to have a better performance than MLRM.

V. CONCLUSION

In this paper, we consider one-to-all broadcast scenarios and put forward a novel GreedyCode based reliable broadcast protocol MCMR-GreedyCode, which is two-fold: channel assignment and link scheduling. Specially, we propose the Level Channel Assignment Strategy (LCAS) algorithm and determine the number of data packets to be sent each time according to the feedback from one-hop neighbor nodes in MCMR-GreedyCode. That is, when one node is about to send data, it makes channel assignment first, then link scheduling. The similar process repeats until all destinations receive complete data. Through simulation, we can conclude that our protocol has a better performance than Pacifier, R-Code, GreedyCode, MCM and MLRM.

However, the OBT in MCMR-GreedyCode is just considered within one hop local area, which is not always optimal. One better greedy strategy is to estimate the efficiency of one node from the respect that how much it will potentially make benefit for all the other nodes that have not received complete data, which is our further work.

ACKNOWLEDGMENT

This work was supported by the Fundamental Research Funds for the Central Universities under grants WK2100100011, Anhui Provincial Natural Science Foundation under grants 11040606M136. The authors would like to thank the editor and anonymous reviewers whose helpful comments improved the quality of this paper. Also we would like to express our gratitude to all the people who ever helped us.

REFERENCES

[1] I. F. Akyildiz, X. Wang, and W. Wang, *Wireless mesh networks: a survey*. Computer Networks, vol. 47, no. 4, pp. 445 - 487, March 2005.

[2] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, *Network information flow*. IEEE Transactions on Information Theory, vol. 46, no. 4, p.1204-1216, July 2000.

[3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, *Trading structure for randomness in wireless opportunistic routing*. in Proc.of ACM SigComm07, Kyoto, Japan, August 2007.

[4] D. Koutsonikolas, Y. C. Hu, and C.-C. Wang, *Pacifier: High-throughput,reliable multicast without crying babies in wireless mesh networks*. in Proc.of IEEE InfoCom 2009, Rio de Janeiro, Brazil, April 2009.

[5] Z. Yang, M. Li, and W. Lou, *R-code: Network coding-based reliable broadcast in wireless mesh networks*. Ad Hoc Networks, vol. In Press, Corrected Proof, pp. 1570 - 8705, 2010.

[6] Xiaobin Tan, Hao Yue, Yuguang Fang and Wenfei Cheng, *Greedy Strategy for Network Coding Based Reliable Broadcast in Wireless Mesh Network*. IEEE Globecom, Anaheim, California, USA, December 2012.

[7] M. Jahanshahi, M. Denghan and M. R.Meybodi, *A mathematical formulation for joint channel assignment and multicast routing in multi-channel multi-radio wireless mesh networks*. Journal of Network and Computer Application, 2011, 34(13), 1869 - 1882.

[8] Guokai Zeng, Bo Wans, Yong Ding, Li Xiao and Matt Mutka, *Efficient Multicast Algorithms for Multichannel Wireless Mesh Networks*. IEEE Trans.on Parallel and Distributed Systems, 2010, 21(13): 86 - 99.

[9] Guokai Zeng, Bo Wang, Matt Mutka, Li Xiao and Eric Torng, *Efficient link-heterogeneous multicast for wireless mesh networks*. Journal, Wireless Networks,Volume 18 Issue 6, August 2012, 605 - 620.

[10] Alicherry M, Bhatia R and Li L, *Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks*. Proceedings of the 11th ACM international conference on mobile computing and networking (MOBICOM 2005), Cologne, Germany, 2005.

[11] Yuan J, Li Z, Yu W, Li B, *A cross-layer optimization framework for multihop multicast in wireless mesh networks*. IEEE Journal on Selected Areas in Communications, 2006.

[12] S. Lee, M. Gerla, and C. Chiang, *On demand multicast routing protocol*. In IEEE WCNC 99, pages 1313-1317, Aug. 1999.

[13] S. Roy, D. Koutsonikolas, S. Das, and Y. C. Hu, *High throughput multicast routing metrics in wireless mesh networks*. In ICDCS 06, 2006.

[14] I.-H. Hou, Y.-E. Tsai, T. Abdelzaher, and I. Gupta, *Adapcode: adaptive network coding for code updates in wireless sensor networks*. in Proc.of IEEE InfoCom 2008, Phoenix, AZ, USA, April 2008.