Digital Communications and Networks 3 (2017) 47–54

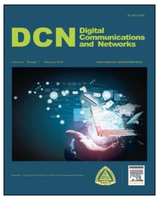
HOSTED BY

ELSEVIER

Contents lists available at ScienceDirect

Digital Communications and Networks

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



Hop-by-hop Content Distribution with Network Coding in Multihop Wireless Networks

Rami Halloush[a], Hang Liu[b,*], Lijun Dong[c], Mingquan Wu[d], Hayder Radha[e]

[a] *Department of Telecommunications Engineering, Yarmouk University, Irbid, Jordan*
[b] *Department of Electrical Engineering and Computer Science, The Catholic University of America, Washington, DC, USA*
[c] *Huawei Research Center, Santa Clara, CA, USA*
[d] *Huawei Technologies USA, Inc., Bridgewater, NJ, USA*
[e] *Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, USA*

A R T I C L E I N F O

Keywords:
Content distribution
Network coding
Multicast
Hop-by-hop transport
Rate adaptation

A B S T R A C T

The predominant use of today's networks is content access and distribution. Network Coding (NC) is an innovative technique that has potential to improve the efficiency of multicast content distribution over multihop Wireless Mesh Networks (WMNs) by allowing intermediate Forwarding Nodes (FNs) to encode and then forward data packets. Practical protocols are needed to realize the benefits of the NC technique. However, the existing NC-based multicast protocols cannot accurately determine the minimum number of coded packets that a FN should send in order to ensure successful data delivery to the destinations, so that many redundant packets are injected into the network, leading to performance degradation. In this paper, we propose HopCaster, a novel reliable multicast protocol that incorporates network coding with hop-by-hop transport. HopCaster completely eliminates the need for estimating the number of coded packets to be transmitted by a FN, and avoids redundant packet transmissions. It also effectively addresses the challenges of heterogeneous multicast receivers. Moreover, a cross-layer multicast rate adaptation mechanism is proposed, which enables HopCaster to optimize multicast throughput by dynamically adjusting wireless transmission rate based on the changes in the receiver population and channel conditions during the course of multicasting a coded data chunk. Our evaluations show that HopCaster significantly outperforms the existing NC-based multicast protocols.

1. Introduction

The predominant use of today's Internet is content access and distribution. Multimedia content traffic is growing at an exponential rate. This trend is expected to continue in the foreseeable future. For example, it is forecasted by Cisco [1] that global mobile data traffic will increase nearly tenfold between 2014 and 2019. Recently, there is renewed research interest in supporting multicast distribution to deliver various services such as live event video streaming, social content pushing, file sharing, software upgrades, mobile TV, as well as other applications for which multiple users concurrently consume the same content [2]. The demand for these applications is becoming increasingly common, and multicast is much more efficient in delivering them than unicast by sharing network resources. The Third Generation Partnership Project (3GPP) recently defined the Evolved Multimedia Broadcast/Multicast Service (eMBMS) standard [3] to support streaming and downloading applications. Several operators have started field trials for eMBMS services.

These multicast applications have strict Quality of Service (QoS) requirements. Many of them require 100% reliability with high throughput. Any packet loss may cause severe quality degradation, and users always desire to get content as quickly as possible. It is challenging to achieve reliable and high-throughput multicast, especially in multihop Wireless Mesh Networks (WMNs) due to interference, channel fading, and limited bandwidth. Furthermore, a unique issue in multicast is bandwidth heterogeneity amongst multicast receivers. The receivers with poor network connectivity or a low-throughput path from the source may greatly degrade the performance of receivers with good network connectivity as the reliability requirements of the worst receiver have to be met.

Traditional reliable multicast protocols, including eMBMS, are client–server based, in which intermediate routers or forwarding nodes simply duplicate and forward packets. These protocols employ end-to-end Forward Error Correction (FEC), Automatic Repeat Request (ARQ) or hybrid FEC-ARQ techniques [4–7] at the application layer of the clients and servers to achieve multicast reliability. However, their

* Corresponding author.
E-mail address: liuh@cua.edu (H. Liu).
Peer review under responsibility of Chongqing University of Posts and Telecommunications.

http://dx.doi.org/10.1016/j.dcan.2016.08.001
Received 8 December 2015; Received in revised form 3 August 2016; Accepted 9 August 2016
Available online 01 September 2016

performance is limited by the multicast receivers with the worst path from the source.

Network Coding (NC) is an innovative technique to improve reliability and throughput in WMNs. The basic idea of NC is to allow intermediate Forwarding Nodes (FNs) to encode data packets, instead of simply replicating and forwarding packets, and thus take advantage of the wireless broadcast medium to reduce the number of required transmissions for delivery of the data [8,9]. Especially, intra-flow random linear network coding [10,11] has attracted interest due to its low control overhead and high efficiency along with implementation simplicity, in which a FN randomly generates linear combinations of received packets belonging to a data flow over some fields, and forwards the coded packets. Random mixing at each FN ensures that if a group of FNs hear the same packet transmission, with high probability, the coded packets generated and forwarded by the different FNs will be linearly independent, removing duplicate packet transmissions over shared wireless medium. A node can not only receive the packets from its direct parent node but also overhear the coded packets of the same data flow transmitted by other neighbors. Intra-flow random NC thus makes opportunistic forwarding and overhearing more effective in WMNs so as to achieve significant performance gains compared to non-coding schemes.

However, the use of NC introduces new challenges in designing a practical multicast protocol. First, a FN does not need to encode and then forward a coded packet whenever it receives a packet from its upstream node because its downstream node may overhear packets from other neighbors. Consider a simple example in which a source S multicasts two packets, P_1 and P_2, to two destination receivers, D_1 and D_2 through a FN, F, as shown in Fig. 1. S transmits two coded packets, $P_1 + P_2$ and $P_1 + 2P_2$ in sequence, which can be represented by the corresponding coding vectors (1, 1) and (1, 2). Assume that F receives both packets, and D_1 and D_2 overhear coded packets (1, 1) and (1, 2), respectively. In fact, F only needs to generate one coded packet from the received two packets, e.g. $3P_1 + 4P_2$, and forward it. D_1 and D_2 can decode and obtain the original packets P_1 and P_2 after they receive coded packet $3P_1 + 4P_2$ from F. However, F may not know that D_1 and D_2 have overheard a packet from S as it has limited knowledge of the reception status of D_1 and D_2 with regard to the two packets sent by S. It is nontrivial for F to decide the number of coded packets it should send and the time when to stop sending. Therefore, one challenge in the NC protocol design is to address how many coded packets each FN should send in order to guarantee all the multicast destination receivers obtain enough packets to decode the original data. In addition, how should the bandwidth heterogeneity of the paths from the source to the different destination receivers be handled by the multicast protocol? To deliver the benefits offered by intra-flow NC, a practical protocol needs to address the above challenges.

Although various NC schemes have been studied under different network settings, practical protocol design for reliable multicast with

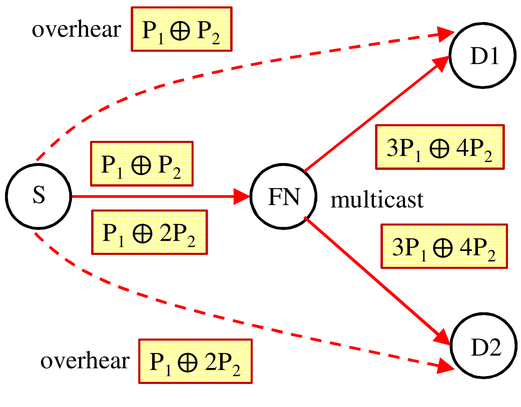


Fig. 1. A NC-based multicast example.

NC in WMNs has received relatively little attention. MORE [11] and Pacifier [12] are the two state-of-the-art intra-flow NC-based multicast protocols with different selections of forwarding node topologies. Both of them employ a transmission credit (*TX_Credit*) approach, in which the source computes and assigns a *TX_Credit* to each FN based on the periodical packet loss rate measurements of the links on the network. The source transmits the coded packets from a data chunk, and the *TX_Credit* values are carried in the packet header that indicates the number of coded packets a FN should transmit upon receiving the packet from its upstream node. An intermediate FN simply determines whether and how many coded packets transmit to its downstream nodes according to the *TX_Credit* assigned to it in the received packet. The successful data delivery is verified through end-to-end acknowledgements (ACKs). The source continues sending coded packets until it receives the ACKs from all its multicast destination receivers. However, with this approach, the FNs may transmit the packets much more than necessary, significantly wasting wireless resources, because it is very difficult to obtain the accurate estimation of the *TX_Credit* for each FN in dynamic wireless environments, and the end-to-end ACKs may be delayed or lost.

In this paper, we take a different approach, and propose a novel intra-flow NC-based hop-by-hop reliable multicast protocol, termed HopCaster, to achieve high-throughput over WMNs and solve the bandwidth heterogeneity issue of multicast receivers. In contrast to the existing NC-based multicast protocols, HopCaster completely eliminates the need for estimating the *TX_Credit,* as well as simplifying multicast management and congestion control. Moreover, a cross-layer rate adaptation mechanism is proposed, which maximizes the multicast throughput by dynamically adjusting the wireless transmission rate to the changes in the receiver population and wireless channels during multicast of a coded data chunk. The evaluation results show that HopCaster achieves significant throughput gains compared to the state-of-the-art NC-based multicast protocols.

The remainder of this paper is organized as follows. Related work is reviewed in Section 2. In Section 3, we present the HopCaster protocol design. Section 4 describes the cross-layer rate adaptation mechanism. In Section 5 we show the evaluation results and performance comparison of HopCaster with the existing NC-based multicast protocol. The paper is concluded in Section 6.

2. Related work

Wireless network coding has been proposed to improve reliability and throughput. Network coding schemes are classified as inter-flow NC and intra-flow NC. With inter-flow NC [13–15], an intermediate FN encodes data packets from different flows, and forwards the coded packets. A receiver decodes the packets to obtain the flow of data targeted to it using its knowledge of another flow. However, in order to obtain inter-flow coding opportunities, the encoded flows need to pass through a common FN with certain network topologies and specific routing [15]. In addition, inter-flow NC typically requires that the sending FN knows what data packets have been buffered or overheard by each of the intended receivers in order to determine how to encode the packets across the flows. This thereby leads to increased control overhead. More recently, it is shown that the use of intra-flow NC with random linear block codes [9,10] can address wireless multicast challenges in an efficient and simple manner because it makes packet forwarding and opportunistic overhearing more effective in WMNs by random mixing of the packets of a flow at the FNs. In [16], a retransmission mechanism with probabilistic network coding was proposed, which used NC to encode multiple packets lost at multiple destinations in one retransmission to effectively recover the lost packets in a multiple-sender multiple-receiver wireless network scenarios. Additionally, a Stackelberg game model was formulated to determine the optimal probability for using network coding to maximize the system performance [17].

There are only a few practical designs and performance studies of intra-flow random NC-based multicast protocols. MORE [11] was the first intra-flow NC-based protocol for reliable unicast and multicast over WMNs, in which any node that overhears the transmission and is closer to the destination may participate in forwarding the packets for a data flow, forming a belt of forwarding nodes, instead of a path towards the destination. The source sends random linear combinations of packets, and a FN also encodes the received packets with random NC before forwarding them. However, belt forwarding can be inefficient, especially for multicast in which multiple belts overlap. Many nodes thus intend to transmit causing a lot of collisions. Pacifier [12] improved upon MORE by using a multicast tree instead of multiple belts for reliable multicast. Only the nodes on the multicast tree perform random network coding of incoming packets and forward the coded packets along the tree, which reduces collisions. It has been shown that Pacifier is able to achieve better performance than MORE [12].

In both MORE and Pacifier designs, the number of coded packets that a FN transmits upon receiving a packet from its upstream node, i.e. the *TX_Credit*, is determined based on periodical measurements of the expected packet loss rate to the destinations, and carried in the packet header by the source. Since the packet loss estimation only represents the expected behavior, the *TX_Credit* cannot guarantee that the destinations will always receive enough coded packets to decode the original data. Therefore, the source keeps transmitting the coded packets from a data chunk until it receives the acknowledgements (ACKs) from all the targeted multicast receivers (a receiver sends the end-to-end ACK after it receives enough coded packets to decode the original data chunk). This is similar to the TCP end-to-end ACK strategy in unicast. There is no cooperation among intermediate FNs and no feedback by the FNs.

Because the source only reacts to the end-to-end ACKs and the FN behavior is controlled via the *TX_Credits* estimated by the source, the above schemes suffer several fundamental problems: (i) Inaccurate estimation of transmission credits may cause injection of redundant packets into the network. (ii) While an end-to-end ACK is being propagated from a multicast destination back to the source, the source will still transmit the coded packets from an already delivered data chunk until receiving the ACK, and these packets will trigger the FNs to send more coded packets, leading to bandwidth waste and more interference. This becomes even worse if the end-to-end ACK gets delayed or lost in the case of congestion and bad link quality. (iii) With many multicast receivers sending acknowledgements to the source, it may lead to ACK implosion [7]. (iv) The source needs to keep sending the packets until the end-to-end ACK from the worst receiver is received, thus it cannot handle heterogeneous receivers well. The proposed HopCaster protocol in this paper uses hop-by-hop acknowledgements and does not need *TX_Credit* estimation. We argue that this approach works better with intra-flow random NC. The preliminary version of our work was presented at a conference [18]. This paper substantially extends it with new figures and discussions. The multicast rate adaptation mechanism is further described. In addition, we largely restructured the manuscript with better presentation and detailed discussions. Peer-to-peer file downloading has been widely studied [19,20], in which peers contribute uploading bandwidth and exchange file chunks. However in P2P systems, peers at the edge of the networks form an application layer overlay with little knowledge of the underlying physical network topology. It is very difficult to take into account the underlying wireless link characteristics such as shared medium and overhearing, and achieve efficient network routing. In addition, most of the P2P applications such as BitTorrent [21] employ TCP for reliable transport between peers. TCP has well-known problems in wireless networks.

Hop-by-hop transport schemes have recently been proposed for the future Internet architectures and content centric networks [22–24]. However, those works mainly focused on the new Internet architecture design and name-based routing protocols without network coding. To the best of our knowledge, HopCaster is the first practical protocol that integrates network coding and hop-to-hop transport for efficient reliable multicast.

3. Hop-by-hop multicast with network coding

The HopCaster design addresses the weaknesses of previous intra-flow NC-based multicast protocols such as Pacifier and MORE, in which the forwarding is based on the *TX_Credit* that is determined by the source, and the source stops sending based on the end-to-end acknowledgements. HopCaster employs a hop-by-hop transport strategy with network coding, and the FNs inform its upstream nodes to stop sending.

3.1. Hop-by-hop coded data transmission

HopCaster builds a multicast tree and leverages it for opportunistic overhearing and network coding in hop-by-hop transport. We will describe the procedures to establish and maintain the tree in the next subsection. As shown in Fig. 2, the solid lines represent the parent–child links on the multicast tree, and the dashed lines indicate that a node can overhear the packet transmissions from its other neighbors such as grandparent, sibling, and child nodes, due to the broadcast nature of wireless medium. We see that there are many opportunities for a node to overhear on the multicast tree.

A large file is divided into multiple chunks at the source before distribution reduce decoding delay at the receivers. A chunk is further divided and encapsulated into k packets. Random linear block codes [10] are applied across the packets belonging to a chunk to generate the coded packets. A coded packet here is a random linear combination of the k original packets with coefficients chosen from a Galois field of size 2^8.

HopCaster uses receiver-driven hop-by-hop transport. Multicast users (receivers) may request a chunk of data by sending REQ messages. The source broadcasts the coded packets of the requested chunk as it is allowed to transmit by the media access control mechanism along the multicast tree. Each transmitted packet is also augmented with its chunk ID and the vector of the random coefficients used to generate the packet.

A FN on the multicast tree receives or overhears coded packets not only from its direct parent but also from any of its neighbors (grandparent, sibling, and even child nodes). After receiving a packet, the FN checks whether the overheard packet is innovative, that is, whether it is linearly independent with the packets obtained from previous transmissions by Gaussian elimination. An intermediate FN on the multicast

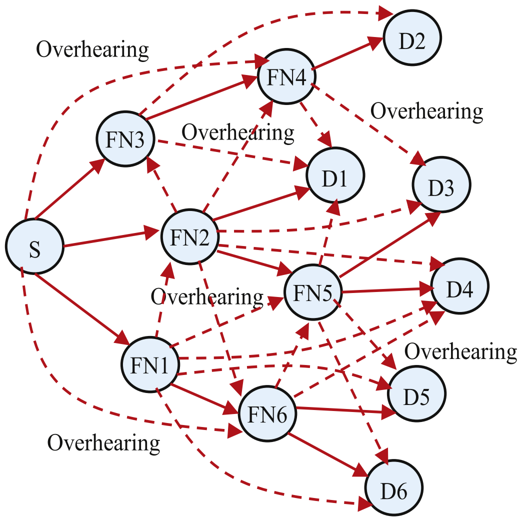


Fig. 2. Overhearing along a multicast tree.

tree caches all the innovative packets of the chunk, and also performs their random linear combinations and sends the re-encoded packets. For random linear NC, the full rank is equal to the number of original packets in the chunk k that the source intends to transmit to the destinations. A node is able to decode the chunk to obtain the original data once it receives the full rank.

In contrast to Pacifier and MORE, data transport in HopCaster operates in a hop-by-hop manner. After an intermediate FN successfully overhears enough innovative packets of a chunk from all its neighbors to reach the full rank, it performs the following operations.

1. The FN sends an ACK-REQ message to its parent. An ACK-REQ message acknowledges that the node has received a data chunk and indicates whether it requests the next chunk.
2. The ACK-REQ message is also used by the FN for controlling the data to it. A FN requests the next chunk only if it has enough buffer space to accept it. There is no need for additional mechanisms for flow and congestion control, which simplifies network management. Note that this is not the same as backpressure-based hop-by-hop flow control [25], where backpressure between adjacent nodes is used for a node to adjust its packet forwarding rate of a continuous flow. In HopCaster, a parent node sends out the packets of a requested chunk as fast as possible (since the requesting children have enough buffer to accept it).
3. The FN becomes a new source for the chunk and is responsible for delivering this chunk to their children.

The one-hop ACK-REQ message is sent unicast from a child to its parent. The link layer such as that in IEEE 802.11 provides certain reliability for unicast. In case an ACK-REQ is lost, it will be retransmitted by the child. The one-hop ACK-REQ is far more reliable, and incurs much less delay than the end-to-end ACK used in MORE and Pacifier. Furthermore, in HopCaster, a FN transmits the coded packets of a chunk by request from its children (receiver-driven), not triggered by receiving a packet from its parent. The advantage is that even if the one-hop ACK-REQs gets lost or delayed, a packet originating from a parent node will not trigger more transmissions from the child nodes (as the case in MORE and Pacifier).

If a chunk has been requested by its children, a FN can start creating random linear combinations of the packets it has received so far for this chunk and broadcasting them without waiting to reach the full rank so as to keep the pipeline going and reduce delay. Random NC removes transmission of duplicate copies of a packet over wireless medium, and the probability that a node receives non-innovative packets from its neighbors exponentially decreases with the code length [10].

A parent node stops transmitting the data chunk once receiving the acknowledgements from all its immediate children. It moves to the next chunk if at least one of its children requests the next chunk. An optimization here is that the parent node can immediately start sending the next chunk as soon as it receives the first ACK for the current chunk and a request for the next chunk from one of its children. The parent node sends the coded packets from a list of requested chunks in a round robin fashion. A chunk is removed from the list after it is acknowledged by all the children. With the later scheduling scheme, the children with good link quality can finish receiving the whole file faster without waiting for their counterparts with bad link quality. The parent stops transmitting after all the chunks are acknowledged by all the children.

A FN is responsible for sending the chunks it has acknowledged to its children. It keeps the received chunks of data in its buffer as long as it is still in the multicast group and has the capacity available in its buffer. It uses the Least-Recently-Used (LRU) policy for its buffer replacement [26], instead of a traditional FIFO queue. The FN at the upstream of a bottleneck link will inform its parent to stop sending a chunk after it has received and cached the chunk. This hop-by-hop control compensates bandwidth fluctuations, pushes data toward the destinations as close as possible, and reduces the impact of multicast receiver heterogeneity. A low-throughput lossy wireless link only affects the receivers behind it. The cached copies may also be used during network topology changes or by new receivers as described below. In case all the data chunks in the buffer are in current use due to very small buffer size, a node can simply delay sending its request to the parent node until the buffer space becomes available.

The source and FNs on the multicast tree maintain a soft-state *chunk request and ACK table*. The chunk request and ACK table contain the information for which chunks requests are pending and whether a REQ or an ACK-REQ message has been received from a child node for a particular chunk. Every parent node knows the list of its children through multicast tree construction.

3.2. Multicast tree establishment and request procedures

HopCaster uses a join scheme to build the multicast tree, and populates and refreshes the chunk request and ACK table. The tree roots at the source, and consists of all the shortest paths from the source to the destination receivers. It is assumed that a unicast routing protocol, such as OLSR [27] or HWMP [28,29], is used to construct unicast paths in the network. Such a unicast routing protocol is needed by other services anyway in WMNs. Our HopCaster protocol is essentially implemented as a shim layer on top of layer 3.

A destination receiver interested in a data file sends a request (*REQ*) message to its parent on the path toward the root of the tree, i.e. the source. The REQ message contains a *chunk ACK field* (*BAF*) that is a bitmap to indicate which chunks of the file have been successfully received by the sending node. The REQ message is processed hop-by-hop along the path toward the source. If a parent node has cached the chunks requested by the children (the corresponding bits in the BAF bitmap field have not been set yet), the requested chunks will be served from this parent node to save bandwidth and reduce delay. The parent node updates the BAF in the REQ message to include its locally cached chunks. It then sends the REQ message to its parent. If an intermediate node can serve all the chunks requested by its children, it does not have to forward the REQ message upstream to its parent node. Otherwise the REQ message propagates toward the source. Note that with HopCaster, a new receiver is efficiently served by the closest node that has cached the requested data, in contrast to MORE and Pacifier in which a joining node will always be served from the source. Furthermore, when a new REQ is received from a child, a parent checks whether there is a match with one of the previous requests from another child. If so, the parent is already in the process of obtaining the chunks requested by this new child. It will not forward this new REQ message and simply waits for the data.

The states in the chunk request and ACK table of the source and FNs are soft and are discarded after a timeout. To provide reliable delivery, a destination receiver is responsible for sending its request periodically toward the source to refresh the state if it still wants the data. For bandwidth efficiency, REQ messages are combined with chunk ACK messages, i.e. using ACK-REQ messages, if possible.

When changes in network topology or link breaks are detected, the path between the destination and the source is repaired or reconstructed by the underlying unicast routing protocol. To speed up the multicast recovery process, the affected nodes are informed of the link break or routing change events. If a node changes its parent node towards the source, the node sends a REQ message to its new parent node on the reconstructed path toward the source. The REQ message contains the BAF to indicate its data delivery progress, i.e. which chunks it has successfully received. If a node loses a child node due to a routing change, it removes the state for this child. In summary, compared to MORE and Pacifier, HopCaster removes the need to estimate the value of *TX_Credit* for each FN, and avoids redundant packet transmissions due to inaccurate *TX_Credit* estimation. It handles the heterogeneous receivers better by allowing a FN at the upstream of a bottleneck link to buffer data for its children. The hop-by-hop ACK and request mechanism reduces the feedback delay and alleviates redundant packets injected into the network.

4. Multicast rate adaptation

Existing radios such as IEEE 802.11 [30] support multi-rate capability at the physical (PHY) layer. Different PHY layer transmission modes use different Modulation and Channel Coding (MCS) schemes, and result in different transmission data rates. It is known that a signal transmitted with a higher rate PHY mode requires a higher Signal-to-Noise Ratio (SNR) at the receiver to successfully decode because it uses a more efficient but less robust modulation scheme along with less channel coding overhead. In other words, at a certain received SNR level, a PHY transmission mode with a higher data rate yields a higher Bit Error Rate (BER). This means that, when the wireless channel is in a good condition, a high data rate transmission mode is more desirable because the receiver will likely receive the signal with a SNR value high enough to correctly decode the signal, and the high data transmission rate will improve the throughput. On the other hand, when the channel is in a bad condition, the SNR at the receiver is expected to be low. Sending the signal using a high data rate transmission mode would most probably lead to errors in decoding at the receiver. A lower rate but more robust mode is thus needed to transmit the signal. Therefore a rate adaptation scheme is desirable, which dynamically adjusts the PHY transmission mode depending on channel conditions in order to improve throughput.

At each forwarding hop in multicast, a parent node may have multiple children. The children with good channel conditions and more overhearing opportunities will obtain sufficient packets and complete its receiving process of a data chunk faster than their poorly connected counterparts. Thus, the population of the children will change during the multicast of a data chunk. A multicast sender can thus adapt its data rate used for transmitting coded packets to the changing population of multicast receivers and their current channel conditions so as to maximize the multicast throughput. However in the existing NC-based multicast protocols, the sender at each hop transmission simply chooses a low robust rate to broadcast the coded packets to multiple receivers, which is not optimal.

Most rate adaptation protocols in the literature [31,32] deal with unicast rather than multicast. Channel conditions are estimated by measuring the received signal strength (RSS) of a feedback message such as ACKs in the MAC layer. However, the IEEE 802.11 standard does not provide feedback messages for multicast packets in the MAC layer. A rate adaptive multicast protocol [33] was proposed, in which some multicast receivers sent acknowledgements to the sender. However, this MAC layer solution requires major MAC and PHY layer changes. Furthermore, it does not consider the changes in the population of receiving nodes in multicast.

The hop-by-hop nature of HopCaster, in which the source and FNs keep track of their children, motivates a cross-layer rate adaptation mechanism to assist HopCaster in determining optimal transmission rate at the physical layer. In this section, we propose a new multicast rate adaptation protocol that leverages the hop-by-hop ACK-REQ messages at the transport layer and takes into consideration the changes in receiver population during multicast.

In our proposed protocol, a sending node selects a PHY mode for a packet transmission to maximize the multicast throughput for its intended children, and adapts to the changing population and channel conditions of its children. Given a radio technology and PHY mode, the packet loss probability is estimated based on channel SNR γ. Just as an example, DPSK, QPSK and CCK modulations with different channel coding schemes are used in IEEE 802.11b PHY transmission modes $m = 1, 2, 3$ and 4 to get transmission rates of 1 Mbps, 2 Mbps, 5.5 Mbps, and 11 Mbps, respectively. The packet loss probability is calculated as [31],

$$P_e(L, \gamma, m) = 1 - [1 - P_e^1(24, \gamma)] \times [1 - P_e^m(L, \gamma)] \tag{1}$$

where $P_e^1(24, \gamma)$ is the error probability of the Physical Layer Convergence Procedure (PLCP) sublayer header that is 24-byte long and always transmitted with PHY mode 1. $P_e^m(L, \gamma)$ is the error probability for the L-byte data payload and MAC header transmitted with PHY mode m that can be expressed as,

$$P_e^m(L, \gamma) = 1 - (1 - P_b^m(\gamma))^{8L} \tag{2}$$

where $P_b^m(\gamma)$ is the average BER corresponding to SNR γ under PHY mode m.

From Eq. (1), it is clear that the packet error probability depends on the PHY transmission mode, the packet size, and the channel SNR γ. Note that this is true for other radios such as 802.11a/g/n/ac even if they use a more complex PHY modulation, channel coding, and antenna techniques in different PHY modes. In general, a relationship database between the packet loss rate and the used PHY mode under different packet sizes and channel SNR values can be built through modeling or offline measurements.

Different children nodes in multicast experience different channels and receive signals with different SNR values from the parent node. Given the received SNR, γ_n, for a receiving node n ($n = 1, 2, \ldots, N$), the value of the packet loss rate, $P_{e,n}(L, \gamma_n, m)$, can be derived for each PHY mode m using the relationship database. Suppose a multicast sender currently has N receiving children nodes, the objective is to find the best PHY mode for the multicast sender to achieve the overall maximum throughput among all its intended receiving nodes. The objective function is formulated as,

$$\underset{m}{\operatorname{argmax}}\left\{\sum_{n=1}^{N} r(m)[1 - P_{e,n}(L, \gamma_n, m)]\right\} \tag{3}$$

where $r(m)$ is the data rate for PHY mode m used by the multicast sender.

Each child node on the multicast tree measures its channel SNR for the packets received from its parent node, and periodically feed the channel SNR measurements back to its parent node. A parent node in HopCaster knows which children still need the coded packets for a particular chunk (based on the chunk request and ACK table). Thus the parent node estimates the objective function and selects the best PHY mode m when transmitting a packet to its children. The SNR reporting messages are combined with the ACK-REQ messages if possible to reduce overhead. Once the parent node receives an ACK-REQ for a chunk from one of its children, the multicast rate adaptation mechanism is informed of this event, and it readjusts its PHY transmission mode to obtain the maximal multicast throughput based on the link qualities of the children that have not completed reception of this chunk.

5. Performance evaluation

To evaluate the performance of HopCaster, we simulate both HopCaster and Pacifier with NS-2 [34], and compare their performance in various settings. We choose Pacifier, instead of MORE, for performance comparison with HopCaster because it is reported in [12] that Pacifier achieves better performance than MORE. We performed our evaluation under many different network topologies and parameter settings. We report results with the following settings. We have a total of 50 nodes. We generate 10 scenarios, each scenario corresponding to a random topology by placing 50 nodes randomly in a 1000×1000 square meter area. There is one data source node and 9 multicast destination receivers requesting the file. The source and destinations are chosen randomly in each scenario. Note that a node could be selected as a destination and a FN at the same time. Furthermore, we simulate a realistic wireless channel with a shadowing propagation model [35] and set the maximum radio transmission range to be 276 m.

In the simulations, each chunk is divided into 32 packets with a 1460 byte payload in each packet. For Pacifier, we follow the implementation in [12]. To compute *TX_Credit* required by Pacifier, nodes periodically exchange hello messages. By keeping track of the lost hello messages, the loss probabilities of all the links in the network are calculated, and then the *TX_Credit*s are estimated. Next we present our simulation results for the cases with static and dynamic multicast trees, and discuss the performance gains attained by HopCaster.

5.1. Static multicast experiments

In these experiments, the multicast tree is established at the beginning of the experiments, and no new receiver requests to join the tree for receiving the data during the course of data transmission. The dynamic cases will be investigated later. Fig. 3 shows the average throughput of Pacifier and HopCaster in the 10 different network scenarios described above, in addition to the overall average throughput over all the scenarios. Here the average throughput for a scenario is obtained by averaging the throughput values over all destination receivers in the scenario. The transmission data rate in those experiments is set to 2 Mbps. As shown in the figure, HopCaster achieves a higher throughput compared to Pacifier in all scenarios. The gain ranges from 6% (scenario 3) to 29% (scenario 9), and the average is 12%. The higher throughput of HopCaster results from injecting fewer NC-coded packets into the network, reducing the total number of required transmissions. This is further clarified in Fig. 4, which shows the number of transmissions by HopCaster and Pacifier in all the above scenarios. Each bar in the graph is divided into two parts: the bottom part corresponds to the number of coded data packet transmissions and the top part corresponds to the number of ACK packet transmissions. Note that in HopCaster, the ACKs are hop-by-hop, and in Pacifier the ACKs are end-to-end. HopCaster results in sending fewer data packets and more ACK packets than Pacifier. The hop-by-hop ACKs in HopCaster significantly reduce the number of redundant data packets injected into the network. Thus, HopCaster has less total number of packet transmissions (data and ACK) in all scenarios except scenario 4 (both protocols have almost the same number of transmissions). In addition, the channel time of sending a data packet is much larger than that of an ACK packet because the ACK packet is much smaller. Fig. 5 shows the redundancy in both protocols. Redundancy is computed as the total number of bytes in all of the transmitted packets (data and ACK) divided by the actual file size in bytes. HopCaster results in much less redundancy over all scenarios.

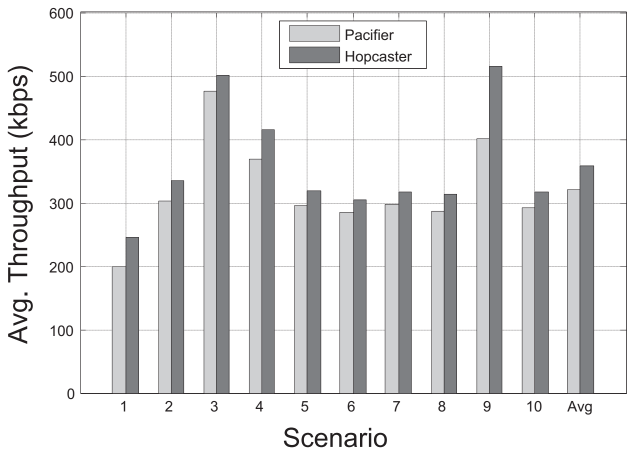


Fig. 3. Average throughput of the proposed HopCaster and the existing Pacifier under 10 different scenarios.

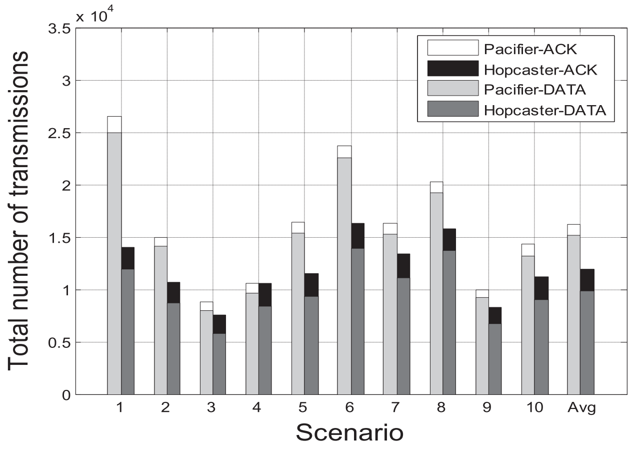


Fig. 4. The number of transmissions (ACK and data packets) of the proposed HopCaster and the existing Pacifier under 10 different scenarios.

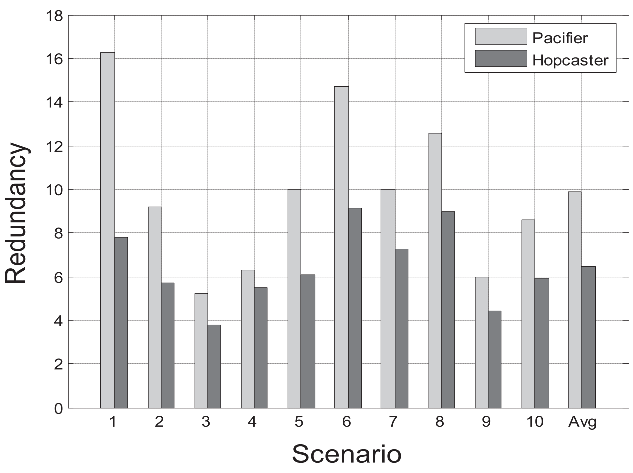


Fig. 5. The number of redundant bytes that are injected into the network by the proposed HopCaster and the existing Pacifier in 10 different scenarios.

5.2. Dynamic multicast experiments

In this subsection, we consider the dynamic multicast cases, in which new receivers request to join the multicast for receiving the data during the course of transmission. Specifically, the simulation starts with one source and 8 receivers, i.e., one of the 9 receivers in the previous experiments will be idle at the beginning of a dynamic multicast experiment. At some point in time, that node will request to join the multicast tree. For HopCaster, the joining node will use the request mechanism described in Section 3.2. For Pacifier, as described in [12], the joining node will send a join message toward the source (the root of the tree) that in turn will register the joining node as a new multicast receiver. We compare the new receiver's throughput under HopCaster and Pacifier in the above 10 different network scenarios. In each of the scenarios, the new receiver and the time of the request will be chosen randomly (the same randomly chosen values will be used for both HopCaster and Pacifier in this experiment). Fig. 6 shows that the new receiver's throughput under HopCaster is much greater than that with Pacifier. The reason is that in HopCaster, the new receiver is immediately served from the closest upstream node in the multicast tree, while in Pacifier, only the source can serve a new node since the source needs to compute the *TX_Credit* for the FNs to send the coded packets to the newcomer. This demonstrates a desirable feature of the hop-by-hop transport approach, which over time, the data will be moved deeper in the multicast tree and closer to the destinations. A new receiver will be served by the closest node that has cached the requested data.

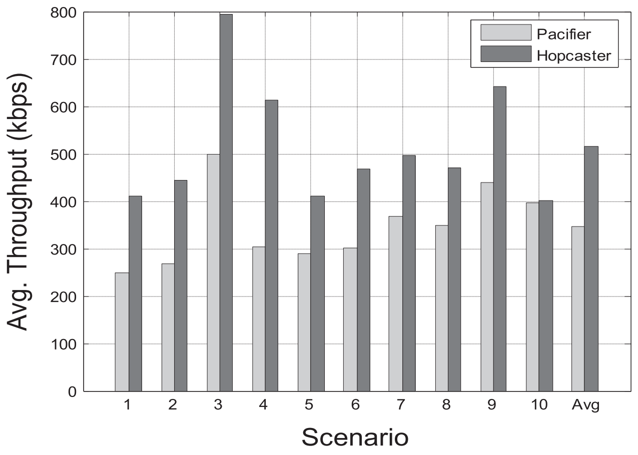


Fig. 6. Average throughput of the new receiver that dynamically joins the multicast with the proposed HopCaster and the existing Pacifier, respectively, in 10 different scenarios.

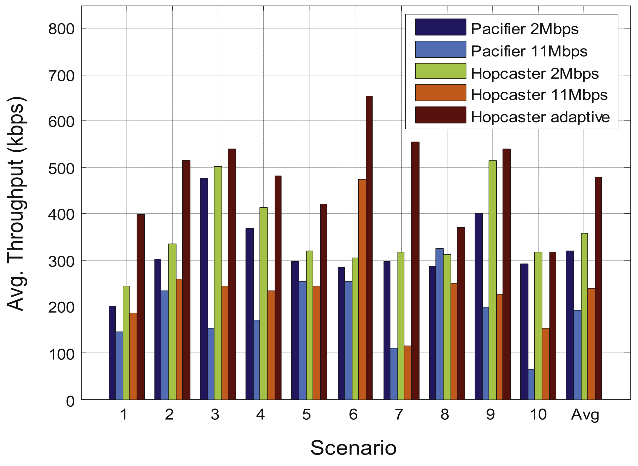


Fig. 7. Average throughput of the proposed HopCaster and the existing Pacifier under 10 different scenarios using 2 Mbps, 11 Mbps, and adaptive multicast data rates.

5.3. Impact of multicast rate adaptation

In this subsection we demonstrate the throughput gain provided by the multicast rate adaptation mechanism described in Section 4. In our simulations, whenever a parent node is about to multicast a packet, it evaluates the optimal data rate according to Eq. (3) for $m = 1, 2, 3, 4$ that correspond to data rates of 1 Mbps, 2 Mbps, 5.5 Mbps, and 11 Mbps, respectively. Eq. (3) is evaluated only for the child nodes that have not acknowledged the current chunk. For comparison, Fig. 7 illustrates the average throughput values for the above 10 network scenarios using the adaptive data rates as well as 2 fixed data rates, 2 Mbps and 11 Mbps. The performance of both HopCaster and Pacifier degrades at 11 Mbps compared to 2 Mbps. This is because of high packet loss rate at 11 Mbps, which requires more transmissions in sending data to the destinations. Furthermore, the ACK-REQ messages may be lost and retransmitted, leading to injection of more redundant packets into the network before switching to the next chunk. Note that the proposed cross-layer multicast rate adaptation mechanism greatly improves the throughput in all the experiment scenarios. The reason is that the rate adaptation mechanism enables a parent node to estimate the current conditions of the wireless links to its intended children. Therefore, it is able to dynamically adjust the data transmission rate to maximize the overall throughput across all its intended children.

6. Conclusions

There are many challenges in designing an intra-flow NC based protocol for efficient reliable multicast over WMNs, including how many coded packets a FN should send and how to handle the bandwidth heterogeneity of multicast receivers. In this paper, we have designed HopCaster, a novel reliable multicast protocol that incorporates intra-flow NC with hop-by-hop transport. Compared to the existing intra-flow NC-based multicast protocols, HopCaster eliminates the need for estimating the number of coded packets each FN should send, avoids redundant transmissions, as well as simplifies multicast management and congestion control. We have also proposed a cross-layer rate adaptation mechanism that enables HopCaster to optimize data transmission rate in hop-by-hop multicast by taking into account the changing population of multicast receivers and the wireless channel variations. Our simulations show that compared to Pacifier, a state-of-the-art intra-flow NC-based multicast protocol, HopCaster greatly reduces the number of required transmissions over the wireless network to deliver multicast data, and achieves higher throughput. Furthermore, we show that the advantages of HopCaster are more prominent in the situation that a new node dynamically requests joining the multicast.

References

[1] Cisco visual networking index: forecast and methodology, 2014–2019 ⟨http://www.cisco.com⟩.
[2] D. Lecompte, F. Gabin, Evolved multimedia broadcast/multicast service (embms) in lte-advanced: overview and rel-11 enhancements, IEEE Commun. Mag. 2012, 50 (11).
[3] 3GPP TS 23.246, Multimedia broadcast/multicast service (mbms); architecture and functional description, rel-13, 2015.
[4] L. Rizzo, Effective erasure codes for reliable computer communication protocols, ACM Comp. Comm. Rev. 1997, 27 (2).
[5] L. Rizzo, L. Visicano, Rmdp: an fec-based reliable multicast protocol for wireless environments, Mob. Comput. Commun. Rev. 1998, 2 (2).
[6] E. Pagani, G. Rossi, Reliable broadcast in mobile multihop packet networks, in: Proceedings of ACM MOBICOM, 1997.
[7] A. Sobeih, H. Baraka, A. Fahmy, Remhoc: A reliable multicast protocol for wireless mobile multihop ad hoc networks, in: Proceedings of IEEE CCNC, 2004.
[8] R. Ahlswede, N. Cai, S.-Y. Li, R. Yeung, Network information flow, IEEE Trans. Inf. Theory , 2000, 46 (7).
[9] S.-Y. R. Li, R. W. Yeung, N. Cai, Linear network coding, IEEE Trans. Inf. Theory , 2003, 49 (2).
[10] T. Ho, M. Medard, J. Shi, M. Effros, D. Karger, On randomized network coding, in: Proceedings of Allerton, 2003.
[11] S. Chachulski, M. Jennings, S. Katti, D. Katabi, Trading structure for randomness in wireless opportunistic routing, in: Proceedings of ACM SIGCOMM, 2007.
[12] D. Koutsonikolas, Y. Hu, C. Wang, Pacifier: high-through, reliable multicast without crying babies in wireless mesh networks, in: Proceedings of IEEE INFOCOM, 2009.
[13] J. Zhang, Q. Zhang, Cooperative network coding-aware routing for multi-rate wireless networks, in: Proceedings of IEEE Infocom, 2009.
[14] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, J. Crowcroft, Xors in the air: practical wireless network coding, in: Proceedings of ACM SIGCOMM, 2006.
[15] B. Ni, N. Santhapuri, Z. Zhong, S. Nelakuditi, Routing with opportunistically coded exchanges in wireless mesh networks, in: Proceedings of IEEE SECON, 2006.
[16] Y. Li, Y. Wu, B. Cao, L. Qiao, W. Zhang, W. Tang, Retransmission mechanism with probabilistic network coding in wireless networks, in: Proceedings of IEEE Globecom, 2014.
[17] B. Cao, L. Qiao, Y. Li, Stackelberg game theoretic approach for probabilistic network coding in retransmission mechanism, in: Proceedings of IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs (MCSoc), 2014.
[18] R. Halloush, H. Liu, L. Dong, M. Wu, H. Radha, Hopcaster: A network coding-based hop-by-hop reliable multicast protocol, in: Proceedings of IEEE GlobeCom, 2012.
[19] S. G. M. Koo, K. Kannan, C. S. G. Lee, Neighbor-selection strategy in peer-to-peer networks, in: Proceedings of IEEE ICCCN, 2004.
[20] Y. M. Chiu, D. Y. Eun, Minimizing file download time over stochastic channels in peer-to-peer networks, in: Proceedings of CISS, 2006.
[21] Bittorrent ⟨http://www.bittorrent.com⟩.
[22] F. Zhang, Y. Zhang, A. Reznik, H. Liu, C. Qian, C. Xu, A transport protocol for content-centric networking with explicit congestion control, in: Proceedings of IEEE ICCCN, 2014.
[23] H. Liu, X. D. Foy, D. Zhang, A multi-level dht routing framework with aggregation, in: Proceedings of ACM ICN, 2012.
[24] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, R.L. Braynard, Networking named content, in: Proceedings of ACM CoNEXT, 2009.
[25] B. Scheuermann, M. Transier, C.L.M. Mauve, W. Effelsberg, Backpressure multicast congestion control in mobile ad-hoc networks, in: Proceedings of ACM CoNEXT, 2007.
[26] J. Jeong, M. Dubois, Optimal replacements in caches with two miss costs, in: Proceedings of ACM Symposium on Parallel Algorithms and Architectures, 1999.

[27] IETF RFC 3626 Optimized Link State Routing Protocol (OLSR).
[28] IEEE 802.11 s Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 10: Mesh Networking, 2012.
[29] D.D. Couto, D. Aguayo, J. Bicket, R. Morris, A high throughput path metric for multi-hop wireless routing, in: Proceedings of ACM MOBICOM, 2003.
[30] IEEE 802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2013.
[31] S. Wong, H. Yang, S. Lu, V. Bharghavan, Robust rate adaptation for 802.11 wireless networks, in: Proceedings of ACM MobiCom, 2006.
[32] J.P. Pavon, S. Cho, Link adaptation strategy for ieee 802.11 wlan via received signal strength measurement, in: Proceedings of IEEE ICC, 2003.
[33] A. Basalamah, H. Sugimoto, T. Sato, Rate adaptive reliable multicast mac protocol for wlans, in: Proceedings of IEEE VTC, 2006.
[34] The network simulator - ns-2 ⟨http://www.isi.edu/nsnam/ns/⟩.
[35] I. Glover, P. Grant, Digital Communications, 3rd edition, Pearson Education, Harlow, England, 2010.