Abstract—Network coding is a network packet transmitting protocol, applied to increase the achievable throughput for the exchange of information between two terminals through one relay. Throughput efficiency this paper reviews latency and bandwidth efficient coding algorithm based on principles of network coding for retransmitting lost packets in wireless multicast network. It outlines the basic principles and important techniques for retransmission of packets in network coding. Network coding is new paradigm that allows the intermediate nodes in a network to create new packets by combining packets received on their incoming edges. The main application of the networking coding technique includes content distribution, peer to peer networks, and wireless ad-hoc networks. Such network is characterized by highly dynamic set of users and frequent topological changes. There are two main advantages of Network coding approach are potential throughput improvements and a high degree of robustness.

Keywords—Network Coding, Bandwidth, Re-transmission Algorithm, Packet Re-transmission, Benefit Algorithm

I. INTRODUCTION

In a traditional packet-switched network, data flows in defined, discrete "pieces" from the source to the destination like corpuscles in the bloodstream. At the transmitting station, the outgoing message is broken into packets, each of which contains some of the message data intact. The packets do not necessarily all travel along the same route but they all eventually arrive at the same destination, where the receiving computer reassembles them into the original message. The main problem with this method is that when the overall network traffic volume is high, bottlenecks are common, resulting in long delays. Packets tend to bunch up at certain nodes, sometimes in excess of the node’s ability to process them. Other routes and nodes may remain under-utilized. Today data travelling over the Internet are much like crates of oranges travelling the interstates in the back of a truck. The data are loaded in at one end, unloaded at the other, and nothing much happens to them in between.

II. NETWORK CODING

Network coding is a method of optimizing the flow of digital data in a network by transmitting digital evidence about messages. The "digital evidence" is itself, a composite of two or more messages. When the bits of digital evidence arrive at the destination, the transmitted message is deduced rather than directly reassembled.

When forwarding packets in a wireless mesh network, airtime can be saved by exploiting the information already present at the destinations of a forward. One way of doing this is Network Coding, where an XOR operation is performed on two packets before transmission. To extract one of the included packets, the destination must simply XOR the network coded packet with the other included packet.

\[(p1 \text{ XOR } p2) \text{ XOR } p2 = p1\]

Fig. 1A simple example of how NC increases the throughput. It allows Alice and Bob to exchange a pair of packets using 3 transmissions instead of 4 (numbers on arrows show the order of transmission)
To give the reader a feel for how COPE works, we start with a fairly simple example. Consider the scenario in Fig. 1, where Alice and Bob want to exchange a pair of packets via a router. In current approaches, Alice sends her packet to the router, which forwards it to Bob, and Bob sends his packet to the router, which forwards it to Alice. This process requires 4 transmissions. Now consider a network coding approach. Alice and Bob send their respective packets to the router, which XORs the two packets and broadcasts the XOR-ed version. Alice and Bob can obtain each other’s packet by XOR-ing again with their own packet. This process takes 3 transmissions instead of 4. Saved transmissions can be used to send new data, increasing the wireless throughput.

With network coding, however, the contents of different packets are mixed together. Given enough information about how the mixing was done, a computer at the receiving end can separate the data back out again. Because each hybrid packet in some sense represents the contents of more than one regular packet, the method can end up saving bandwidth. Network coding was able to consistently increase the data capacity of a wireless network to about three times what it was initially.

Until the hybrid packets are decoded, however, they’re total gibberish, and without some basic information about how they were produced, decoding them is next to impossible. So, while network coding offers a more efficient use of bandwidth, it also offers an efficient means of securing information.

III. ALGORITHMS FOR NETWORK CODING

The algorithms for network coding to differentiate between different retransmission techniques for network coding.

1) Sort-by-Time

This algorithm is greedy in nature. Packets are sorted according to their arrival time with the first packet being the one that arrives the earliest. Packets are sorted once all the NAK arrive at the sender. Every time the sender starts with the first packet in the queue, and iteratively combines with subsequent packets in the queue as long as the combined packet can be decoded (i.e., all the receivers of the combined packet already have all but one packets in the combined packet). It waits until the end of the batch to start retransmission.

Sort by time gives optimum solution

2) Sort-by-Utility

The coding algorithm which E. Rozner et. al. proclaims to be the delivering the best performance for a one-hop multicast network is sort by utility. CUk is the in a sort by utility coding algorithm, the Tx first transmits N packets, and then sorts the packets in descending order of their CUk values, using arrival time as tie-breaker for those packets have equal packet utility.

Once the packets are sorted, the remaining operation of Sort-by-Utility is essentially a greedy coding algorithm, i.e. the Tx then iteratively starts coding successive packets starting from packets having highest packet utilities and codes them with successive sorted packets as long as the coded packet can be decoded by all receivers.

<table>
<thead>
<tr>
<th>Table I Transmission Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATRIX</td>
</tr>
<tr>
<td>R1</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
</tr>
<tr>
<td>C3</td>
</tr>
<tr>
<td>C4</td>
</tr>
<tr>
<td>C5</td>
</tr>
</tbody>
</table>

Consider as an example the matrix given in Table I. Sort-by-Utility algorithm after initial packet transmission (c1 to c5) would sort the transmitted packets based on its packet utilities CUk, i.e. c2, c1, c4, c3, c5. Then Tx transmits the packets as follows: c2, c1⊙ c3, c4, c5. Thus, requiring a total of 4 retransmissions with an average time to decode a packet to be 4.4 time slots.

IV. BENEFIT ALGORITHM

BENEFIT unlike Sort-by-Utility does not need to wait until the end of the batch size (N packet transmissions) before starting the retransmission process. It start transmitting coded packet once the prospective coding packets satisfy
the following three conditions: CodingBenefit(), ColumnsBenefit() and CombinationBenefit() (see Table II). Retransmitting as soon as the right conditions are met rather than wait till the end of the batch size in effect reduces the time to decode the packet. BENEFIT works on the basis that it is not necessary for the coded packet to be decodable by all the receivers immediately, assuming that the non-decodable coded packet can be decoded based on future transmission of coded packet(s). This principle is in essence the key strength of BENEFIT and thus, this way it contrasts the traditional packet coding rule.

A) CodingBenefit()
Checks the following equality:
DecodeBenefit() ≥ MinimumBenefit()

B) DecodeBenefit()
 Finds out how many receivers STA will benefit immediately from the transmission of the coded packet, 0 ≤ DecodeBenefit() ≤ M

C) MinimumBenefit()
From the list of prospective coding packets, find cK with minimum current packet utility cuK , 0 ≤ MinimumBenefit() ≤ M

D) ColumnsBenefit()
From the list of prospective coding packets, checks if every packet can be decoded by at least one receiver STA immediately.

E) CombinationBenefit()
From the list of considered packets for coding, calculates the number of receiver STA which will benefit either immediately (ru1=1) or in future (ru2 ≥ 2) from the coding of the prospective coding packets. And then checks if it’s equal to DesiredBenefit(see Fig 1).run value in the context of CombinationBenefit() is computed only for the packets in pros pk[ ], CombinationBenefit() ≥ DecodeBenefit().

F) DecodeSearch()
Decodes the arrived coded packet if it can. If the packet gets decoded, then search the memory for any previously non-decodable packet which can now be decoded based on the current decoded packet, and decode it.

G) Benefit immediately immediate Coding
The decoding of coded packet on the spot, without need for information from future transmission Consider the algorithm given in Fig. 1, illustrated with the transmission matrix given in Table I. After the Tn, transmit c1, c1 is stored as the first prospective coding packet. Tn then transmit c2, c1 and c2 are then checked for CodingBenefit() and ColumnsBenefit() conditions, which they satisfy. Hence they are then checked for CombinationBenefit() condition. Since both c1 and c2 satisfy the CombinationBenefit() condition as well, the packets are coded and transmitted. Only R1 is not able to decode c1⊕c2 immediately (current value of cu1=cu2=1). The algorithm then scan the next packet c2 and stores it as the first prospective coding packet, and the Tn then transmit c3. Since c2 and c3 satisfy CodingBenefit() and ColumnsBenefit() conditions but not CombinationBenefit() condition, c3 is therefore saved as a prospective coding packet. The Tn then transmits c4, which in addition to c2 and c3 satisfy CombinationBenefit() condition. Hence the packets are coded and transmitted. All receivers are able to immediately benefit from the transmission of c2⊕c3⊕c4.

Decoding of coded packet at R1 after decoding c2⊕c3⊕c4, decodes c1⊕c2 using c2 and obtains c1. The last packet in the batch c3 is then transmitted and stored as prospective coding packet, however as it is obvious, there is not any possibility of finding coding packets for c3, the algorithm decrements the value of DesiredBenefit twice, following which c3 is retransmitted without any encoding in the third scan cycle.

V. EXHAUSTIVE SEARCH

We develop an exhaustive search algorithm to minimize the number of retransmissions. This algorithm is computationally very expensive and is not for practical use. Instead it serves as an interesting baseline comparison to quantify the effectiveness of the other coding algorithms. First, we introduce a few notations. Let M denote the number of packets required for retransmissions. Let S denote a state, indicating for each packet i which nodes need it and which
nodes have it, namely (N(i), H(i)). The exhaustive search algorithm first generates all possible packet combinations. There are 2M packet combinations, since each packet either belongs to a packet combination or not. The goal is to find a smallest number of packet combinations that converts the current state to the state where every node gets the packets it needs (i.e., N(i) = {} for every i). To identify a minimum set of packet combinations, we build the following coding tree. The root of the tree is the current state. Starting from the root, we try every packet combination. A packet combination is considered useful if it allows at least one receiver to get a packet it needs if there is no loss. For each useful packet combination, we add a child node to the root; we also label the edge of to the child with the packet combination and label the node with the state after all nodes receive the packet combination. Packets combinations that are not useful are simply ignored. After going through all the packet combinations, we then repeat the process – for each of the child nodes we identify the useful packet combinations and add them to the next level of the tree. The process continues until we reach a state where every node gets the packet that it needs. The depth of the tree at that node is the minimum number of transmissions required (assuming the depth of a root is 0). Moreover, the packet combinations marked along the path from the root to that node are the set of packets to transmit that minimizes the number of transmissions.

**Conclusion**

- sort by time sorts by time sorts by utility sorts by utility value cuk.

the above graph demonstrates the results for 10 receivers as large no of receivers increase the computational complexity, for small number of receivers of sort by time even outperforms exhaustive search. But if the number of receivers is increased than sort-by-utility continues to perform the best. In all cases, the ratios are lowest under low packet loss rates because the packets lost at different receivers are more likely to be different under low loss rates and create more coding opportunities.

- BENEFIT requires a total of only 3 retransmissions (c₁ ⊕ c₂, c₂ ⊕ c₃ ⊕ c₄ and c₅) in contrast to 4 retransmissions used by Sort-by-Utility, with an average time to decode a packet of 1.9 time slots (c₁ ⊕ c₂ transmitted after the transmission of c₂, and c₂ ⊕ c₃ ⊕ c₄ transmitted after the transmission of c₄) in contrast to 4.4 time slots used by Sort-by-Utility.
BENEFIT outperforms Sort-by-Utility both in terms of
- Retransmission
- Bandwidth and
- Packet delay.

While the bandwidth performance of BENEFIT and Sort-by-Utility is almost similar for low loss probability and/or small network, for such networks BENEFIT can still be useful for real-time applications which are highly delay sensitive. However, both these techniques will come at a tradeoff cost of an increase in retransmission ratio. Flexibility to balance throughput-delay trade off in BENEFIT allows the network designer to modify the algorithm based on the network requirements.

The exhaustive search gives the best result in all cases but has high latency due to computational complexity. Its computational complexity is exponential.

The complexity of sort by time \( n^2 \) in worst case and \( n \) in best case.

The complexity of sort by utility \( n^2 \) and in best case \( n \).

The exhaustive search best and worst case \( 2^m \), \( m \) being the no of packets in queue.

VI. LITERATURE REVIEW

1. "An Efficient Network Coding based Retransmission Algorithm for Wireless Multicast," Jalaluddin Qureshi, Chuan Heng Foh and Jianfei Cai: In this paper we study a latency and bandwidth efficient coding algorithm based on the principles of network coding for retransmitting lost packets in a single-hop wireless multicast network and demonstrate its effectiveness over previously proposed network coding based retransmission algorithms.

2. "Wireless Broadcast Using Network Coding Dong Nguyen," Tuan Tran, Thinh Nguyen, and Bella Bose: In this paper we study some network coding techniques to increase the bandwidth efficiency of reliable broadcast in a wireless network. Our proposed schemes combine different lost packets from different receivers in such a way that multiple receivers are able to recover their lost packets with one trans-mission by the sender.

3. "ER: Efficient Retransmission Scheme for Wireless LANs," Eric Rozner, Anand Padmanabha Iyer, Yogita Mehta, Lili Qiu, and Mansoor Jafry: study an efficient retransmission scheme (ER) for wireless LANs. Instead of retransmitting the lost packets in their original forms, ER codes packets lost at different destinations and uses a single retransmission to potentially recover multiple packet losses.


5. "Network Information Flow," Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung: We study a new class of problems called network information flow which is inspired by computer network applications, considering a point-to-point communication network on which a number of information sources are to be multicast to certain sets of destinations.

VII. CONCLUSIONS

Various algorithms can be used as packet retransmission. With each algorithm with its own benefits and drawbacks exhaustive search is most efficient in terms of number of retransmission. Benefit algorithm with most efficient compute time. Sort by utility and sort by algorithms can be used for their easy implementation, but are considerably inefficient as compared to benefit and Exhaustive search in terms of number of retransmission.

REFERENCES


