DACK-XOR: An Opportunistic Network Coding Scheme to Address Intra-flow Contention over Ad Hoc Networks

Radha Ranganathan[1], Kathiravan Kannan[2], P. Aarthi[3], and S. LakshmiPriya[4]

[1] Asst Professor, IT Dept, Easwari Engineering College, Chennai, India 600089
[2] Dean, Easwari Engineering College, Chennai, 600089 India
[3] Software Engineer, Philips India Ltd, Bangalore, 560045 India
[4] Student, Arizona State University. Tempe, AZ 85287-1003
{radhupriya@yahoo.co.in,
{kathirraji,aarthi.npk,slplakshmipriya}@gmail.com}

Abstract. Network coding is a novel technology that exploits the intrinsic broadcast nature of wireless media, to significantly reduce the number of transmissions (hops). Our primary focus is to minimize intra-flow contention by exor-ing TCP-DATA and TCP-ACK packets belonging to the same TCP flow, ensuring that the packets are never delayed in the process. Network coding always comes with the overhead of intermediate nodes having to buffer packets so as to successfully perform decoding. This requires the nodes to maintain large buffers. We also propose a new technique in which we retain only the last delivered packet in the buffer. However, at times when the same node gets access to the medium repeatedly, keeping only the last sent packet in the buffer will not suffice. Hence we introduce a rate based Cross-layer Transport Solution (CLTSP) that inserts a delay interval (called out -of-interference delay) between two consecutive packets transmitted. This reduces intra-flow contentions by leaps and bounds. Our unique combination of using a Network Coding technique along with a suitable TCP variant improves the throughput gains with significant improvement in handling medium contention by reducing the number of transmissions.

Keywords: Network Coding, Intra-flow Contention, DATA-ACK in TCP, Ad Hoc Networks, EXOR coding, Pseudo-Broadcast.

1 Introduction

With gadgets galore, people are living a highly networked life these days. The Internet and the cell-phone network has literally transformed the life of every single individual on this planet. The growing number of users coupled with the increasing demand for newer technology is the principle factor that drives the network scientists to think beyond the horizon.

In the Internet, the responsibility for directing data traffic lies with special-purpose devices called routers. Internet service providers monitor the flow of traffic across their networks and, if they spot congestion, revise the routers' instructions accordingly. With the cell network, two people a block apart could be having a phone conversation, but they aren't directly exchanging data. Rather, they're sending data to a cell tower that determines what to do with it. In recent years, many network scientists have turned their attention away from centralized networks — such as the Internet and the cell-phone network — and toward ***ad hoc networks***, wireless networks formed on the fly by.

The Transmission Control Protocol is a well-known de-facto protocol in developing today's internet. Due to its wide acceptance and deeper understanding, it is desirable to extend and adopt its functionalities in Wireless Networks also. But studies have shown that TCP performs poorly in MANETs [1]. The reasons can be attributed to the typical behavior of the nodes in the wireless medium - mobility, high bit error rate, unpredictability, contentions, long connection times, etc. Some TCP-Specific problems identified over Mobile Ad-hoc networks[2]:

- TCP misinterprets route failures as congestion
- TCP misinterprets wireless errors as congestion
- Intra-flow and inter-flow contention reduce throughput and fairness
- Delay spike causes TCP to invoke unnecessary retransmissions
- Inefficiency due to the loss of retransmitted packet.

We propose to focus our work towards reducing the intra-flow contention that has significant impact on the throughput of the network. Of the various means of alleviating this kind of medium contention pertaining to a single connection, our work uses a new kind of Network Coding technique to address the issue.

The core idea of Network Coding is to mix several packets together for transmission, thereby greatly increasing the amount of packet information transmitted in a single hop. The intermediate routing nodes no longer just store and forward packets. They also process the packets before transmission. This processing done at the relay nodes helps to reduce the number of packet transmissions at the MAC layer and consequently the number of times a node contends for the medium.

2 Related Work

Network Coding techniques are being touted as networking's next revolution [3] and it is considered as a paradigm shift in data transportation across networks. No wonder that the academia and the Industrial giants are looking forward to embrace this technology in the existing network infrastructure systems. Since the time when Network Coding Theory [4] stemmed out of the lineage of Information Coding theory, lot of pioneering work has been carried out to exploit the Network Coding paradigm with multi-faceted approaches. A variety of coding techniques were adopted for different purposes. Network Coding has huge potential impact in multicast applications as illustrated in the works of [5], [6]. The Unicast traffic scenario has been considered in the [7], that exploits physical layer capabilities and in COPE [8] wherein Medard et al. have proved the prowess of the Network Coding abilities with a practical test-bed implementation. This work looks out for effective coding opportunities to forward multiple packets in a single transmission. However, this demonstration could not produce desired levels of throughput gains in TCP end-to-end connections. Sundararajan et al. [9] proposed a intra-flow random network coding scheme with a new interpretation of ACK packets with the concept of Degrees of Freedom. But this work failed to bring out the fullest potential of the wireless network coding as it employed coding operations only in the end hosts and not in every relay node. Chuan Qin et al. [10] proposed a first-of-a kind system to integrate both inter and intra flow network coding operations in the wireless medium. But the suitability of this project in TCP connections is quite ambiguous. A Mac-level Network coding scheme named – Piggycode [11] had exclusive focus on the intra-flow packet coding mechanism which significantly improves TCP performance. Mario Gerla et al. used this work in their ComboCoding technique [12] with an additional timer mechanism to the Piggy code system. Both these works emphasize the need for an almost infinite buffer for effective operations Our proposed work, DACK-XOR, which is DATA-ACK EXOR Network Coding Scheme|| is to enhance the pure coding-decoding procedures, i.e. XOR- network coding in a TCP connection to mix TCP DATA and TCP ACK packets and strive to achieve better throughputs and reduce the stress on the medium contention issue by restructuring buffer systems and Transport Layer mechanisms.

3 Network Coding - A Stress on the Buffer?

Network Coding mandates the maintenance of an infinite decoding buffer to store packets that have already been sent, in order to perform successful decoding. This comes as a huge overhead in terms of the buffer space required. Also, this introduces a significant delay in the decoding process because the packet needs to be searched for in the infinite buffer. Therefore, we resort to a technique which retains only the last packet delivered, in the buffer. This takes the stress off the buffer to a large extent.

However the above mentioned technique works only if there is a strict fairness in medium access. So we make use of a rate based TCP, to ensure that the same node does not repeatedly get access to the medium. This TCP variant, in addition to reducing intra-flow contention also sees to it that packets do not get clogged at a specific node. Fig. 1 shows a sample schematic view of our DACK-XOR ing Scheme of Network Coding.

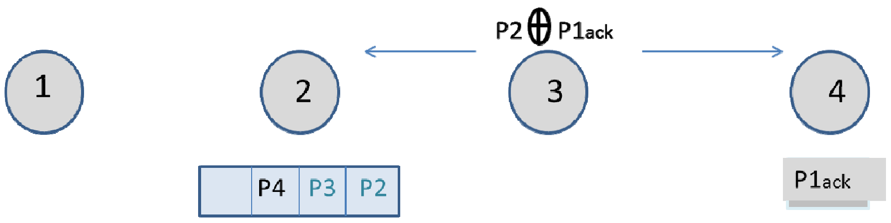

Fig. 1. Network coding illustration with new proposal

4 System Architecture

The network coding logic is integrated into the MAC layer. Packets coming from the upper layers are lined up in the outgoing queue, waiting to be transmitted. Packets that are received from outside are also queued up, waiting to be forwarded. The network coding layer looks for a coding opportunity by checking if a TCP-DATA residing in the queue can be exored with a TCP-ACK and vice-versa. If the look up is successful, the two packets to be exored are sent into the encoder where the TCP-ACK packet is appended with an appropriate number of zeroes so as to be exored with TCP-DATA packet. The coded packet is then broadcast. A copy of the sent packet is kept in the decoding buffer (packet pool). When a coded packet is received, it is sent into the decoding unit. Using the unique packet identifier as the search key, the native packet is retrieved from the packet pool. This known packet is exored with the received packet to obtain the original packet. Fig.2 is our Generic node architecture that supports DACK-XOR Network Coding technique.

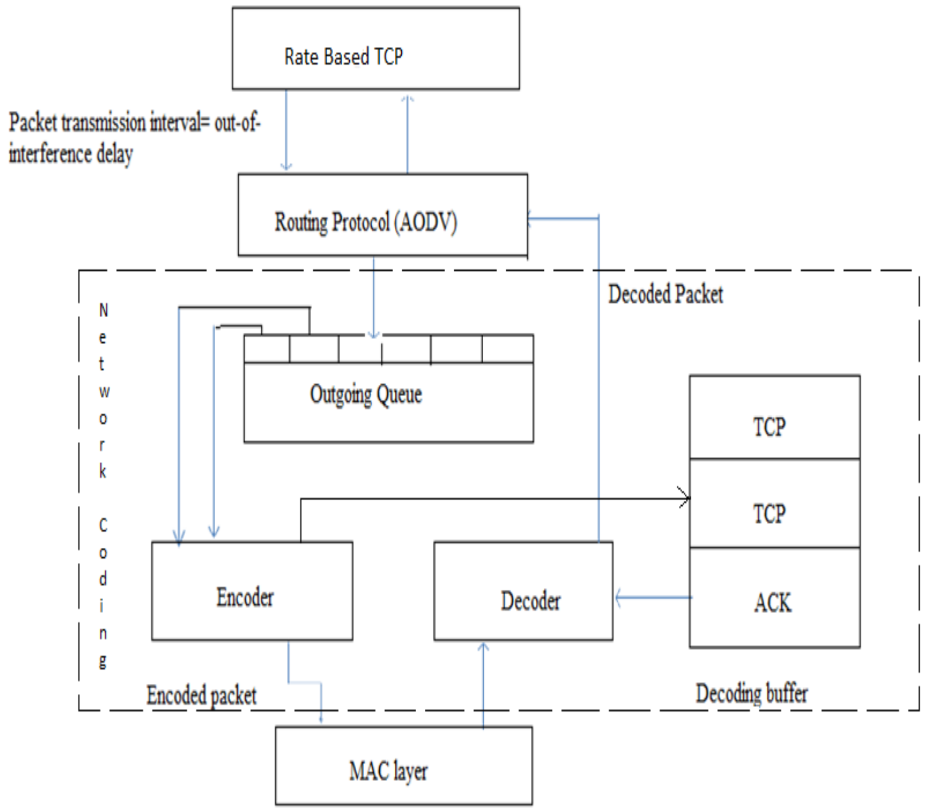

Fig. 2. Generic Node Architecutr to support DACK-XOR

5 Module Description and Design

Our implementation aimed at lessening the medium contention within a specific TCP flow by reducing the number of transmissions of packets across the medium involves three different modules. The first one is the heart of the Coding-Decoding part of the Network Coding Module. The second one is focused on improvising the Encoded packets' transmission by using a pseudo-broadcast mechanism instead of the usual

Broadcasting approach used in many of the Network Coding works, as this offers better reliability and feedback mechanism. The third and the last module works on refreshing the packet pool appropriately to reduce the stress on the buffer. This work looks out for effective coding opportunities to forward multiple packets in a single transmission. However, this demonstration could not produce desired levels of throughput gains in TCP end-to-end connections.

5.1 Piggy Code Network Coding

In this module, we implement network coding which piggy-backs data and acknowledgement in a single packet transmission to two different receivers in a single TCP connection, assuming that TCP DATA and ACK packets travel in the same path but in opposite directions. Here, we make use of decoding buffers of infinite size to store a copy of all the sent packets.

1) Opportunistic Encoding Process:
When we de-queue the packet at the head of the interface queue at the relay node, we check if there are coding opportunities, by searching for a packet of opposite type. We are said to arrive at a coding opportunity when a packet of opposite type with interchanged source-destination pair addresses and belonging to the same flow are identified in the queue. The Algorithm for Interface Queue Management is given in Table. 1.

Table 1. Algorithm for Interface Queue Management

```
INPUT: Q, the Packet Queue at Link Layer
OUTPUT: Packets to be sent to Down Target - MAC
PROCEDURE:
Initialize chained:=0,
     P:=head of Interface Packet Queue ,
CurrPtype:= P • type, CurrConnID:=P • flowID,
CurrSrc:=p • IPsrcAddr, CurrDst:= IpdstAddr
OppType:= (CurrPtype==TCP-DATA)? TCP-ACK : TCP-DATA
If CurrPtype == TCP-DATA or TCP-ACK and chained ==0 then
  pkt:= P
  foreach pkt:=pkt • next
    if pkt!=NULL then
      if pkt•type==OppType and pkt • flowID == CurrConnID
      then
        if pkt•IPsrcAddr==CurrDst and pkt•IPsrcAddr ==
        CurrSrc then
          chained:=1
          remove(pkt)
          p • next:=pkt
          break
        endif   endif  endfor endif
```

If the queue lookup is successful, the two packets to be exored are chained together. The chained packets are de-queued together from the Link Layer Packet Queue and transmitted to the down-target, i.e., the MAC layer.

It is in the MAC layer wherein we perform the actual Network Coding Operation. We EXOR the payload fields of the MAC layer and put them up in the new EXOR-ed packet. The packets arriving at the MAC layer to be passed on to the Wireless Physical Layer are stored in the Packet Pool for decoding purposes. They pool is refreshed at appropriately, as explained in this section later.

We then include a CODED field in the Network Coding header (NCHeader) of the new packet to indicate that the packet is coded. We also insert the sequence ids of the two packets that have been coded. The new packet is now sent further down for transmission. The algorithm used for the Encoding of suitable packets opportunistically, is illustrated in a flow-chart in Fig. 3.

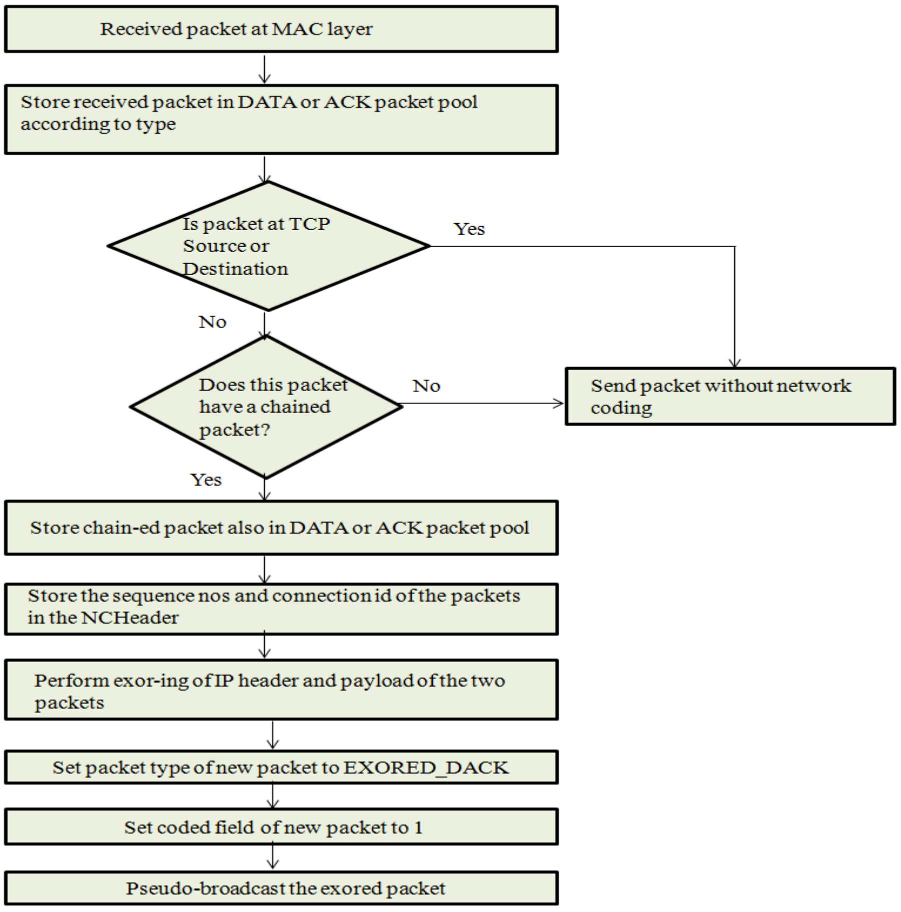
Received packet at MAC layer
Store received packet in DATA or ACK packet pool according to type
Is packet at TCP Source or Destination
Yes
No
Does this packet have a chained packet?
No
Send packet without network coding
Yes
Store chain-ed packet also in DATA or ACK packet pool
Store the sequence nos and connection id of the packets in the NCHeader
Perform exor-ing of IP header and payload of the two packets
Set packet type of new packet to EXORED_DACK
Set coded field of new packet to 1
Pseudo-broadcast the exored packet

Fig. 3. Network Coding - Encoding Process flow chart

2) Decoding Process:

When a coded packet is received, we check if one of the packets inside the coded packet has already buffered been in the pool. This lookup is performed using the packet identifiers as the search key. If the un-coded packet is found, it is exored with the received packet in order to retrieve the original packet. Fig. 4 depicts this process as a flow-chart.

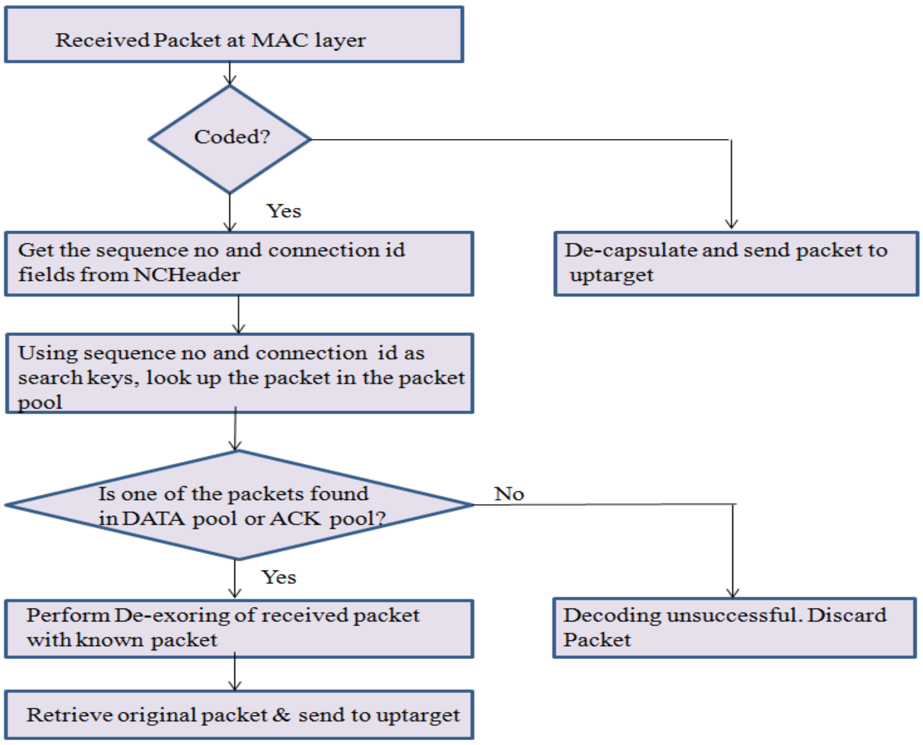
Received Packet at MAC layer
Coded?
Yes
Get the sequence no and connection id fields from NCHeader
De-capsulate and send packet to uptarget
Using sequence no and connection id as search keys, look up the packet in the packet pool
Is one of the packets found in DATA pool or ACK pool?
No
Yes
Perform De-exoring of received packet with known packet
Decoding unsuccessful. Discard Packet
Retrieve original packet & send to uptarget

Fig. 4. Decoding Process flow chart

5.2 Pseudo – Broadcast of Exored Packet

The broadcast of exored packets might lead to collisions in the medium due to lack of RTS-CTS exchange in 802.11. This results in the TCP source having to retransmit the packet. Hence we resort to pseudo-broadcast of the coded packet. The coded packet is now unicast to only one of the two intended recepients by including the receiver address in the MAC header. The other receiver's address is stored in our new NCHeader. Since any node within the interference range of the sender can receive packets that are not destined for it, the node receives the packet and checks the NCHeader to see if it is the other intended recepient. If yes,it performs de-exoring of the packet. Pseudo-broadcast decreases the collision rate to a considerable extent

5.3 Coding Buffer Enhancement – Last Delivered Packet

In this module, we restructure the Piggy Code Buffer of every node to retain only the last packet delivered. This greatly reduces the buffer look-up time for decoding packets at the receiver side. In order to be able to perform decoding successfully, this needs to be done in conjunction with the third module “Rate based transport solution”.

5.4 Implementing Rate-Based Transport Solution

We make use of a rate-based Cross-Layer Transport Protocol (CL-TSP) that enforces strict fairness in medium contention, by making sure that the same node does not get repeated access to the medium. This is achieved by exploiting the spatial reuse of the wireless channel. The CL-TSP inserts a delay in between successive packet transmissions. The delay is calculated by averaging the out-of –interference delay in the forward and reverse path. Essentially, once the CL-TSP layer of a node transmits a packet, it has to wait for a specific amount of time before it can transmit the next one. This prevents intra-flow contentions among successive packets.

We deliver only one data packet at a time from transport layer of source to prevent intra flow contention among them. The time interval between the successive deliveries should be calculated such that it should not be large which may otherwise underutilize the bandwidth of the network and it should not also be small which can otherwise lead to contention among them. If packet 1 is delivered at time t1 and packet 2 is to be delivered at time t2, then the time interval (t2-t1) must be carefully selected by considering the above factors.

We can say that the time interval between the deliveries of successive packets at the transport layer can be the four hop transmission delay which is named as out of interference delay. So, (t2-t1) should be at least the out of interference delay to prevent contention between pkt1 and pkt2.

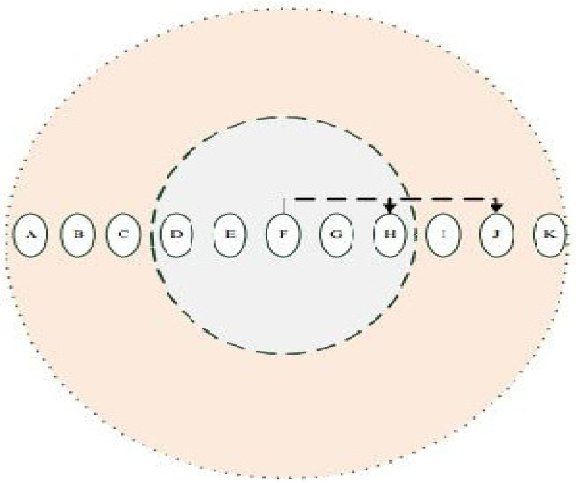

Fig. 5. Analysis of number of hops on out of interference range

We may also argue that the out of interference delay cannot be always represented as four hop transmission delay. It will increase when the nodes are closely placed. For example, we consider Fig.6. Where in the distance between the nodes are taken as 100m (closely located). The transmission range of node F covers node G, H and its interference range covers up to node K. The routing protocol like AODV chooses shortest path which is based on hop count to reach the destination. The next hop for node F is chosen as node H not the node G. The path may be chosen as node F ->node H->node J ....So, we assume that four hop transmission delays can represent the out of interference delay in most cases.

We need to analyze whether this four hop out of interference delay will be sufficient to determine the inter packet delivery period at CL-TSP source. The transport layer should be reliable in nature, the CL-TSP source should receive acknowledgment for every data packet from the end receiver. Also, the forward and reverse path of the transport connection is same in most of the network. In this scenario, if four hop transmission delay alone is considered in calculating the inter packet delivery, it will lead to severe contention with the reverse acknowledgments. So, we need to consider the four hop transmission delay of both data and acknowledgment packets in determining the inter packet delivery period. The total delay for a packet i can be measured as given in Eq. 1

Total delay(i) = max(interference delay values recorded)

We will be taking into account the total delay of recent n packets for calculating the exponential mean which will be represented as the inter delivery time period between the next two packets.

6 Experimental Results

We evaluate the performance of our DACK-XOR Network Coding in a phased manner to emphasize the significant performance improvements gained through our work. The specialty of our work lies in the fact that the modularized parts can be incrementally deployed in real time scenario. To prove this point, we have chosen to evaluate our work with four different variants:

- IEEE 802.11 at MAC layer + TCP New Reno at Transport layer protocol and this combination is named as " original MAC" in the comparison charts.
- IEEE 802.11 at MAC layer + PiggyCode Network Coding Module +TCP New Reno at Transport layer protocol and named as "Piggy Broadcast" in the comparison charts.
- IEEE 802.11 at MAC layer + Piggy Code Network Coding Module with Pseudo broadcast + TCP New Reno at Transport layer protocol and named as "Pseudo Piggy" in the comparison charts.
- DACK-XOR integrated module at MAC + rate based CLTSP at transport layer and named as "Integrated TSP" in the comparison charts.

The finally integrated DACK-XOR package comprises all the modules together – PiggyCode Network Coding with Pseudo Broadcast implementation and a Rate-based

CLTSP solution - to achieve the ultimate objective of alleviating Intra-flow Contention. We have used Network Simulator 2.34 for our Experimental Analysis. Our chosen Network Scenario is a Linear-chain topology with 8 nodes (7 hops). The distance between any two contiguous nodes is set to 220 meters and the transmission range of each node is 250 meters. We consider a single TCP connection with Node 0 as the source and Node 7 as the sink. The various Simulation parameters for our Network Scenario are listed down in Table 2 as shown below.

Table 2. Simulation Setup

Application Type	FTP
Data Rate	Discrete variants from 1Mbps to 5 Mbps
Basic Rate of 802.11 MAC	1Mbps
IFQ Capacity	50
Queue Type	DropTail / PriQueue
Propagation Model	TwoRayGround
Antenna Type	Omni Antenna

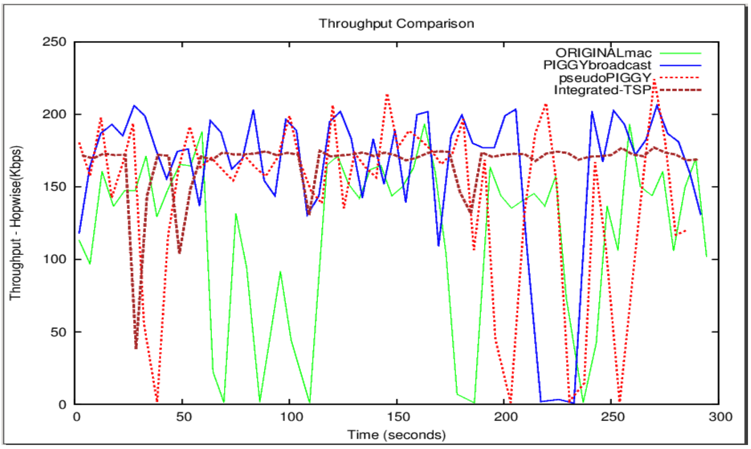

Fig. 6. Throughput variance across time for our Network under different Protocols

6.1 Throughput vs Time

We have shown the performance of our network-coding variants by measuring the throughput at regular intervals of time during the entire simulation. It can be seen from Fig.6 that throughput of our integrated network coding solution is consistently high as opposed to the original scheme without network coding in which the throughput drops to zero many a times.

6.2 Number of Hops vs Throughput

Having shown the continuous throughput variation with time, we next evaluate the same parameter with respect to the number of hops in the network scenario. The graph in Fig. 7 shows the average throughput obtained for varying number of hops. The effectiveness of our scheme can be realized only for a number of hops greater than 5 when the effect of network coding becomes pronounced. Also, the out-of-interference delay comes into play only when the number of hops is greater than 4.

6.3 Throughput under Different Data Rates

Here we measure the average throughput of the four schemes at different data rates such as 1, 2, 3, and 5 Mbps. It can be seen that the effectiveness of network coding cannot be felt at 1Mbps data rate. This is because in spite of TCP trying to inject as much packets as possible, the available bandwidth is less and the number of packets travelling across the medium is also less as explained in [5]. So, only a few packets can be actually encoded (low coding opportunities) at the intermediate nodes. Hence the throughput benefits get nullified by the coding overheads, causing an initial degradation of performance.

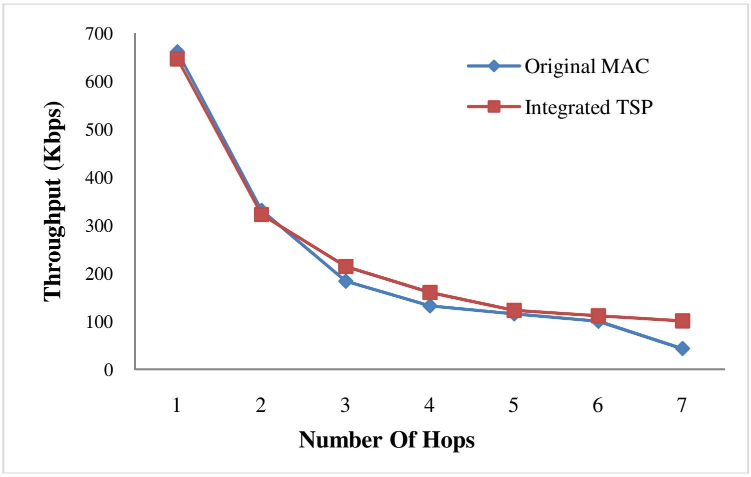

Fig. 7. Average Throughput VS Number of Hops

The effectiveness of the proposed DACK-XOR is realized as the available bandwidth of the network increases. As soon as the link capacity increases, the TCP sender tends to fill it by injecting more and more packets into the network thus increasing the overall coding probability. This implies that the number of saved transmissions increase and, consequently, a further portion of the link capacity results now available to the TCP sender.

6.4 RTS Transmissions

We measure the intra-flow contention by the ratio of RTS Transmissions to Packets Generated. The values shown in Table 3 depict that there is a decrease of RTS transmissions. Our DACK-XOR network coding scheme combines the DATA and ACK packets and sends them in one transmission, thereby reducing RTS ratio. The table shows the relative comparison of the percentage of RTS transmissions when compared with the number of actual packets generated during our simulation.

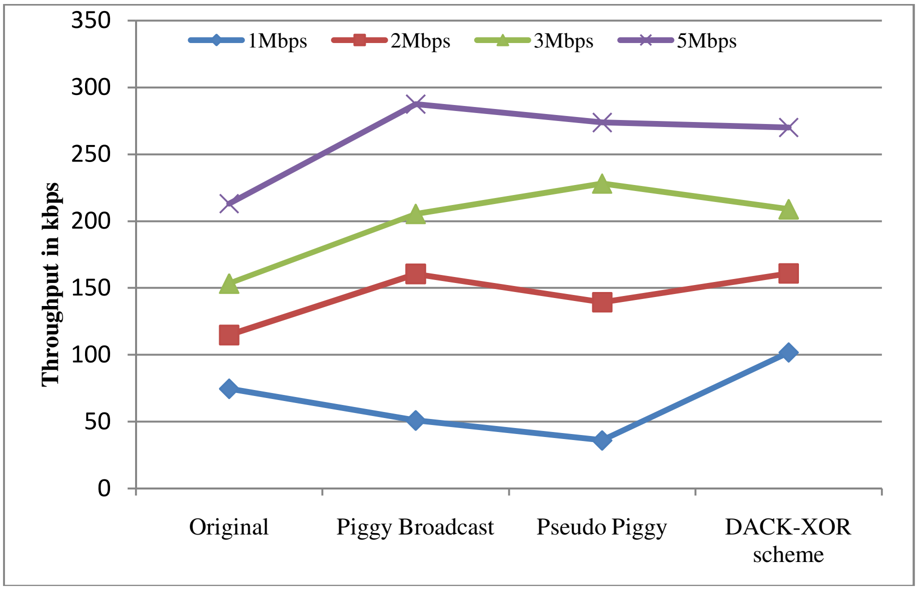

Fig. 8. Average Throughput for Different Variants for Different Data Rates

Table 3. Number of bytes generated and RTS count for Different Variants

Description	Generated Bytes	RTS Transmission	Ratio of RTS TX to Gen. Bytes
Original 802.11 MAC	4360000	85758	1.967%
Piggy code with Broadcast	6208000	111421	1.79%
Piggy code with Pseudo Broadcast	5308000	99256	1.87%
Integrated TSP	6015000	96648	1.61%

7 Conclusion and Future Work

This work presents an improvised network coding scheme that addresses the issue of intra-flow contentions. We reduce the number of packet transmissions by combining DATA and ACK packets travelling in opposite directions, performing a pseudo broadcast, and at the same time, take the stress off the decoding buffers, with the help of the cross-layer, rate based TCP which prevents packet clogging- A condition that is most likely to occur when the same node repeatedly wins medium contention. Although we have optimized the network coding buffer by retaining only the last packet delivered, achieving a decoding probability of 1 is not possible. On an average, for every 8000 TCP packets sent, we have 1 decoding failure. Moreover, we have realized our design by making changes to the TCP layer in addition to those at the MAC layer. Our future work would be to achieve buffer optimizations for network coding at MAC level alone instead of using a TCP variant to indirectly improve the efficiency of network coding.

References

1. Nage, T., et al.: TCP-Aware Network Coding with Opportunistic Scheduling in Wireless Mobile Ad Hoc Networks. In: Consumer Communications and Networking Conference (CCNC). IEEE (2010)
2. Siva RamMurthy, C., Manoj, B.S.: Ad hoc Wireless Networks: Architecture and Proto-cols
3. Network World – Network coding: networking's next revolution? `http://www.networkworld.com/news/2007/121007-network-coding.html`
4. Ahlswede, R., Cai, N., Li, S.-Y.R.: Network Information Flow. IEEE Trans.Inform. Theory (2000)
5. Deb, S., Effros, M., Ho, T., Karger, D.R., Koetter, R., Lun, D.S., Medard, M., Ratna-kar, N.: Network coding for wireless applications: A brief tutorial. IWWAN (2005)
6. Chen, C.-C., Oh, S.Y., Tao, P., Gerla, M., Sanadidi, M.Y.: Pipeline Net-work Coding for Multicast Streams (Invited Paper). Proc. of the ICMU (2010)
7. Wu, Y., Chou, P.A., Kung, S.Y.: Information Exchange in Wireless Networks with Network Coding and Physical-layer Broadcast, MSR-TR (2004)
8. Katti, S., Rahul, H., et al.: Xors in the air: Practical wireless network coding. IEEE/ACM Trans. Netw (June 2008)
9. Sundararajan, J., Shah, D., Medard, M., Mitzenmacher, M., Barros, J.: Network Coding Meets TCP. IEEE INFOCOM (April 2009)
10. Qin, C., Xian, Y., Gray, C.: MIX: Integration of Intra-Flow and Inter-Flow Wireless Network Coding. In: Proc. of IEEE SECON Work-Shops (2008)
11. Scalia, L., Soldo, F., Gerla, M.: PiggyCode: a MAC layer network coding scheme to improve TCP performance over wireless networks. IEEE GLOBECOM (2007)
12. Chen, C.-C., Chen, C., Oh, S.Y., Gerla, M., Sanadidi, M.Y.: ComboCod-ing: Combined Intra/Inter-Flow Network Coding for TCP over Disruptive MANETs. J. Adv.Res. 2(3) (2011)