Maximizing Throughput of SW ARQ with Network Coding through Forward Error Correction

Farouq M. Aliyu, Yahya Osais, Ismail Keshta, Adel Binajjaj
Department of Computer Engineering
King Fahd University of Petroleum and Mineral Resources
Dhahran, Saudi Arabia

***Abstract*—Over the years, several techniques for improving throughput of wireless communication have been developed in order to cater for the ever increasing demand of high speed network service. However, these techniques can only give little improvement in performance because packets have to be delivered as is. As such researchers have begun thinking outside the box by proposing ideas that require relay nodes to temper packets' contents in order to improve the throughput of a network. One of the state of the art techniques in this field is called Network Coding (NC). NC is a state of the art technique that allows relay nodes linearly combine two or more packets in a way they can be recovered upon reaching their destination. However, increasing packet size increases possibility of error affecting it. In this paper, the authors decide to investigate whether adding data recovery technique can improve the performance of a network that uses network coding, if it can, by how much can it? Is it worth the trouble? In order to answer these questions, the authors carried out a quantitative analysis of throughput in a Stop-and-Wait Automatic Repeat reQuest (SW-ARQ) data transmission system with Network Coding (NC) and Forward Error Correction (FEC). Vandermonde matrix is chosen as the coding technique for this research because it has both NC and data recovery characteristics. Python programming language is used to develop three Discrete Event Simulations: SW-ARQ without any NC, SW-ARQ with NC and SW-ARQ with NC and FEC. The obtained results show that SW-ARQ with NC and FEC is superior to traditional SW-ARQ in terms of throughput, especially in channels with high error rates.**

***Keywords*—Network Coding; Automatic repeat request (ARQ); Stop-and-Wait (SW); Vandermonde Matrix**

I. Introduction

Non-ideal behavior of communication channel causes received data to sometimes change from its original form, thus leading to misinterpretation. Automatic Repeat reQuest (ARQ) is one of the basic error control protocols used to provide reliable communication between two wireless devices. There are three main ARQ systems namely; Go-back-N, Selective Repeat and Stop-and-wait [1].

In stop-and-wait (SW) ARQ, transmitter sends a frame and then waits until it receives a reply (i.e. Acknowledgment (ACK) or Negative ACK (NACK)) for the transmitted packet from the receiver. Although, SW-ARQ is simple to implement and guarantees that packets are received in order, it is not efficient because of the wasted time during waiting for replies. Thus, it has low throughput [2]. Throughput can be defined as the average rate of successful data transmission over a network, and it is normally given in bits per second (bps). Go-back-N ARQ and selective repeat ARQ protocols manage to reduce ACK overhead by transmitting multiple frames without waiting for their ACK. For further information on ARQ protocol refer to [1] [2].

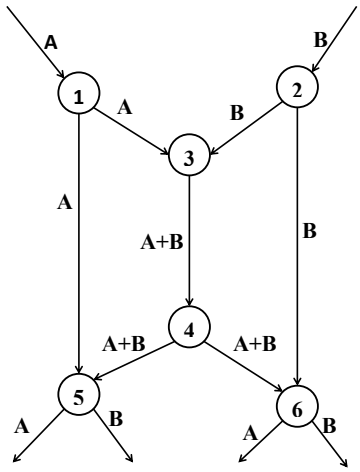

Fig. 1: Butterfly Network Coding in a multicast network

Other data transmission techniques for improving overall network's throughput are Network Coding (NC) and Forward Error Correction (FEC). NC was invented by Ahlswede *et al* [3] in 2000. It is an in-network data processing technique that allows intermediate nodes to aggregate two or more packets into one before forwarding it [4]. Figure 1 shows a butterfly wireless network with the vertices representing routing nodes and the edges representing the path of the packets. NC has several advantages; studies have shown that NC enhances the overall network throughput [1][5], it increases performance in multi-rate networks [4] and increases robustness of the network [6].

Forward Error Correction (FEC) on the other hand, is an error control technique where redundant data is systematically generated and embedded in a packet such that it can be regenerated in the event of error during transmission [7]. In FEC, the transmitter takes N data symbols and encode them with M parity symbols to form $N + M$ new symbols before transmitting them. At the receiver the original data symbols can be reconstructed as long as N out of the $N + M$ received symbols are error free. For more information on FEC refer to [7].

This paper extends the work in [8], where the throughput of a SW-ARQ with NC was investigated. Here we investigate the effects of NC when combined with FEC on the throughput of SW-ARQ using Python high level programming language. The remaining part of this paper is as follows: Section II reviews some of the related work in ARQ and NC. Section III provides detailed information on the developed network model and compares it with the work in [8] where necessary. In Section V analysis of the results obtained and comparison with results in [8] is carried out. Finally, in Section VI conclusion(s) was/were drawn based on the results that were obtained.

II. Related Work

De Vuyst al. in [9] present an analysis of the SW-ARQ protocol, in which it was pointed out that errors occur in bursts as packets are transmitted from the transmitter to the receiver and that the probability of receiving an erroneous packet depends on state of the channel when the packet was transmitted. Gilbert *et al* [10] divide these states into two; GOOD state and BAD state — using two-state Markov Chain to model the channel. Their result shows that the delay (which is inversely proportional to throughput) increases sharply with increase in the capacity of the channel. In order to compensate for the drop in throughput due to increase in capacity, techniques like Forward Error Correction(FEC) and Network Coding (NC) are often used.

In [11], the authors investigate the effect of network coding (NC) on throughput of the three basic ARQ systems. Their findings show that NC significantly improves the throughput of all the three ARQ protocols. Furthermore, studies show that wireless networks using NC give better throughput even though the complexity of the system increases. A novel Automatic Repeat reQuest (ARQ) system for cooperative wireless networks is introduced by Antonopoulos *et al* [12] in 2011, where cooperative and network coding techniques are combined in order to enhance the system's performance. They are able to obtain 85% bandwidth improvement due to the reduction of the total number of transmissions. Li *et al* [13] propose system for wireless broadcast system based on the random network coding using two-State Markov Chain to model the channel. The authors analyze the throughput of a typical SR-ARQ using; Linear Network Coding (LNC) and Random Network Coding (RNC) and it is found that random network coding gives better throughput, especially in the case of a system with a large number of receivers. Liu *et al* [14], introduce NC-ARQ system based on Two-State Markov channel for Cognitive Radio (CR) broadcast, where lost packets are XORed by CR base-station forming a new packet. The new packet is then broadcasted by CR base-station to all of CR users. The results show noticeable improvement in throughput, especially when there are large numbers of CR users. On the contrary, authors in [15] study and analyze the steady-state throughput of SW-ARQ with NC using finite state machine, results obtained show that as the number of incoming links to the base-station increase a bottleneck in information delivery is formed.

Alsebae *et al* [8], study the effect of network coding (NC) system on the throughput of SW-ARQ. The system is simulated using MATLAB SimEvents Discrete Event simulation toolbox. The system is described in Algorithm 1. It measures the throughput of two nodes with similar function to node 3 and 4 in Figure 1. In their work, the transmitter waits for n packets, which are then converted into an $n \times n$ Vandermonde matrix (see Equation (1)). Each row (or block) of the matrix is considered as a new packet. These new packets are then transmitted to the receiver where the original n packets are regenerated. The researchers conclude that SW-ARQ with NC has better throughput, particularly in cases where the channel has high error rate. Finally, they postulated that it could give higher throughput than that which they have accomplished. This led to the research reported in this paper, where FEC abilities of Vandermonde matrix have been exploited. This approach is inspired by the fact that probability of error increases exponentially with increase in packet size. Therefore, there is need to add FEC to the protocol in order to increase its throughput as we shall see in Section III.

Algorithm 1: Pseudo-code for SW ARQ-NC used in protect [8]

```
pktCount ← 0
clear buffer
while (pktCount < n) do
    wait for packet
    buffer ← packet
    pktCount ← pktCount + 1
end while
Matrix ← generated Vodermonde Matrix(buffer)
i ← 0
while (i ≤ pktCount) do
    reply ← transmit(Matrix[i])
end while
```

III. SW-NC with Forward Error Correction

$$V_{(nxn)} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \dots & \alpha_n^{n-1} \end{pmatrix} \quad (1)$$

The proposed system is simulated using two nodes, node 3 and 4 of Figure 1. The system has a transmitter which collects n packets. It converts the n packets into a $n \times n$ Vandermonde Matrix. The matrix takes the general form shown in Equation 1, where each row is a block of data that can be transmitted as a packet. At the receiver, the blocks are broken down into their original components (i.e. the earlier n packets).

Algorithm 2 provides a general overview of the proposed SW-ARQ with NC and FEC. The system consists of three main simulation components namely;

A. Transmitter Model

This part is modeled to contain two sub-modules namely *packet generation* and *encoding*. Packet generator, generates packets at a fixed rate λkbps. Once the time (i.e. $\frac{packet\_size}{\lambda}$) has pass, the packet generation sub-module will add a new data packet to the transmitter's buffer. Every data packet is saved in the format: [seqnum, payload], where seqnum is the packets sequence number and payload is the size of the packet in

Algorithm 2: Pseudo-code for proposed system

```
pktCount ← 0
clear buffer
while (pktCount < n) do
    wait for packet
    buffer ← packet
    pktCount ← pktCount + 1
end while
Matrix ← generated Vodermonde Matrix(buffer)
i ← 1
while (i ≤ pktCount) do
    reply ← transmit(Matrix[i])
    if (reply = NACK and i ≠ pktCount) then
        i ← i + 1
    else
        if (reply = ACK) then
            exitWhile
        end if
    end if
end while
```

bits. This process is repeated until the total number of packets generated equals to a certain number of the pre-programmed packets (n).

After the required numbers of packets were generated, the transmitter forwards them to the Vandermonde Matrix Encoder. There, the packets are stripped off of their headers and trailers before they are converted into blocks representing the rows of the Vandermonde matrix as shown in Equation 3. The Vandermonde Matrix is formed purely from the packets payload. The advantage of Vandermonde Matrix is that the encoded data packets are linearly independent. Hence the receiver is able to recover the packets. The mathematical equation representing how packets are converted to Vandermonde Matrix is as follows:

let $P_1, P_2, \dots P_n$ be packets generated and $b_1, b_2, \dots b_n$ be the blocks generated then,

$$\begin{aligned} b_1 &= P_1 + P_2 + \cdots + P_n \\ b_2 &= P_1^2 + P_2^2 + \cdots + P_n^2 \\ &\vdots \\ b_{n-1} &= P_1^{n-1} + P_2^{n-1} + \cdots + P_n^{n-1} \end{aligned} \tag{2}$$

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ P_1 & P_2 & \dots & P_n \\ P_1^2 & P_2^2 & \dots & P_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ P_1^{n-1} & P_2^{n-1} & \dots & P_n^{n-1} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{3}$$

Note that the first row of the Vandermode Matrix is always filled with ones as such this row is never transmitted. On the contrary, it is transmitted in [8]. This can be seen as an overhead since this information is redundant.

After each transmission, the transmitter pauses until it receives the ACK or NACK from the receiver. On one hand, if the transmitter receives NACK, it transmits the next block in the hope that a different column might be in error, thereby allowing the receiver to apply forward error correction. On the other hand, if the block in question is the last block, then the transmitter keeps sending it (since there are no more blocks to send) until it receives an ACK. As a rule of thumb, transmission is over whenever ACK is received, because it signifies that all packets can be recovered from the blocks received. However, the authors in [8] fail to take the advantage of this unique characteristic of the Vandermonde matrix. Thus, all columns are transmitted.

B. Channel Model

In order to simulate corrupt frames accurately, the channel is modeled using *Binary Symmetric Channel* (BSC) [16][17]. BSC is an independent and identically distributed (i.i.d.) channel with the probability of a given bit "*flipping*" as ϵ also known as *Bit Error Rate* (BER). Therefore, the probability of finding an error in a given frame can be represented by Equation 4.

The channel module keeps checking for the presence of data. Once data is placed on the channel, the channel checks it first; if data is an ACK from the receiver it is passed to the transmitter directly without going through the error simulation module, because ACK/NACK packets are so small in size that error has negligible effect on them as shown by Equation 4. However, if the packet is not an ACK/NACK, the channel passes it to the error sub-module, where *Monte Carlo* method is applied to it in order to randomly choose the blocks to be in error based on the frame error rate equation (i.e. Equation (4)). Frame Error Rate (FER) is the probability that one or more bits in a frame are in error.

$$FER = (1 - (1 - BER)^k) \tag{4}$$

Where, FER = Frame Error Rate
BER = Bit Error Rate
k = Number of bits in a frame

C. Receiver Model

Finally, the packet reaches the receiver which hands it to a sub-module called the *Error Checker*, where the received packet is checked for errors; if error(s) is/are found, then the columns of the Vandermonde Matrix collected so far are checked. If all elements of a column are in error (as in Equation 6) then the original packets cannot be recovered. The receiver is notified and it sends a NACK packet to the transmitter, but saves the corrupt packet in the hope that it can help in future error corrections. Corrupt packets are discarded and new packets are requested in the system proposed by [8].

$$V_{(3x3)} = \begin{pmatrix} 1 & 1 & 1 \\ Error1 & 6 & 3 \\ 4 & 36 & Error2 \end{pmatrix} \tag{5}$$

$$V_{(3x3)} = \begin{pmatrix} 1 & 1 & 1 \\ Error1 & 6 & 3 \\ Error2 & 4 & 36 \end{pmatrix} \tag{6}$$

In a nutshell, receiver attempts forward error correction on the Vandermonde Matrix, if that fails the transmitter then sends the next block of data. However, if it is the last block the transmitter keeps sending it until an ACK is received. Equation 5 and 6 have illustrate scenarios where original packets can and cannot be recovered respectively.

D. FEC technique

Vandermode matrix can be used in conjunction with other error detection techniques to develop a packet recovery system [18]. As shown in Equation 2 and 3, packets (P) from other nodes are encoded into Vandermonde matrix. Each row (also known as Block (b)) is transmitted across the network as a packet. At the receiver these blocks are checked for errors before the original packets are finally recovered.

Suppose an element in the x row and y column of the Vandermonde matrix is represented by $\alpha_{x,y}$, then the nth packet encoded can be retried using Equation 9.

$$P_n = \alpha_{2,n} \tag{7}$$

$$\alpha_{2,n} = \sqrt[x-1]{\alpha_{x,n}} \quad where \quad x > 1 \tag{8}$$

$$\Rightarrow P_n = \sqrt[x-1]{\alpha_{x,n}} \quad where \quad x > 1 \tag{9}$$

IV. Simulation

Although Matlab Simevents was used in [8], Python programming language was used in this research. It allows the programmer limitless flexibility and levels of conception. However, in order to ensure accuracy the system proposed by [8] was first reproduced and its codes was later tweaked to develop our proposed system.

Discrete Event Simulation (DES) approach was used [19][20]. The events used are described in Section III. Three separate codes were written base on the network setup shown earlier in Figure 1. The first code simulates the network without any network coding technique added. The second simulates the network with Vandermonde matrix used as means of coding the network. While the third code uses the Vandermonde matrix as both network-coding and data recovery technique. Each of the three codes were then simulated to thirty seconds of simulation time. The results where then printed in the form of graphs that are presented in Section V, this was done with the help of the "matplotlib" python library [21]. The variables were also exported using the programs IDE and further analysis was carried out on the data. This is possible because Python(x,y) IDE was used for the simulation [22].

Table I enlists parameters used in the simulation of the SW-ARQ communication system with and without NC. These parameters are exact replica of those used in [8], which allow us to compare the performance of the two simulations. However, it is worth noting that packet generation rate (λ) is changed from 50 to $100 packets/s$ in order to ensure maximum performance for all three networks as indicated by Figure 7.

V. Discussion

In this section the throughput performance of the proposed system is presented and analyzed. The section also draws out some possible applications of the proposed system.

TABLE I: Parameters used in simulation

Parameter	Value
pkt_size	1000 bits
lamda (λ)	100 Packets/s (packet generation rate)
Rate	10 Mbps (The system bit rate)
Tprop	15 ms
Tsim	40000 ms (Simulation time)
FER	Forward error rate (values used: 0, 0.1, 0.6, 0.9)
ack_waiting_time	default setting, 0
n	5 blocks per code

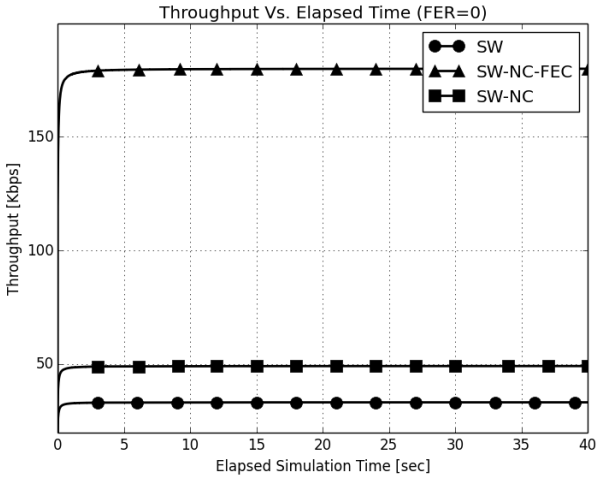

Fig. 2: Throughput of SW, SW-NC and SW-NC-FEC in Error free channel

A. Performance Analysis

For the sake of clarity: SW represents Stop-and-Wait Automatic Repeat reQuest; SW-NC represents Stop-and-Wait Automatic Repeat reQuest with Network Coding which is used in [8]; while SW-NC-FEC represents Stop-and-Wait Automatic Repeat reQuest with Network Coding and Forward Error Correction, which is the proposed system in this paper.

Over an error free channel, the maximum achievable throughput for SW, SW-NC and SW-NC-FEC are shown in Figure 2. It is clear that network coding with forward error correction is superior. This can be attributed to the fact that in an error free channel only one transmission is required in order to transmit all the n packets. In the system developed in [8] however, all packets have to be transmitted mindless of whether earlier sent packets have been received successfully. For SW-ARQ, the throughput of the system is around 33 kbps. This is expected because the time required to transmit a block containing n packets in SW-NC-FEC is the same time used by SW-ARQ in transmitting a single packet.

As the frame error rate (FER) increases the delay in transmission of packet increases, hence the difference in performance (i.e. throughput) as shown in Figure 3, 4 and 5. To investigate the severity in drop of performance with increase in FER, a graph of throughput for the three simulations against their channel's FER (See: Figure 6) is plotted. From the graph, it can be seen that fall in throughput is more obvious in the case

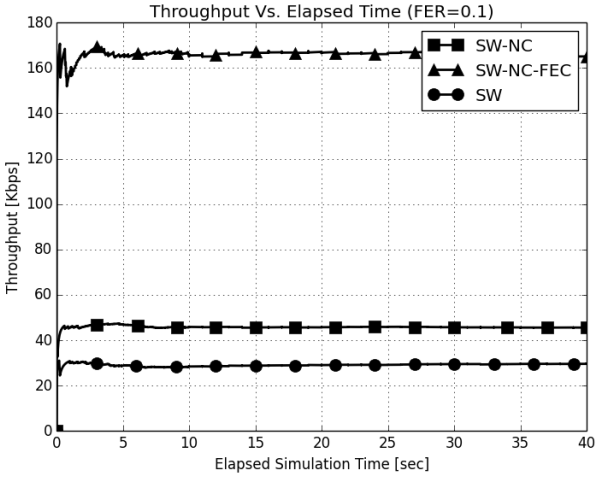

Fig. 3: Throughput of SW, SW-NC and SW-NC when FER=0.1

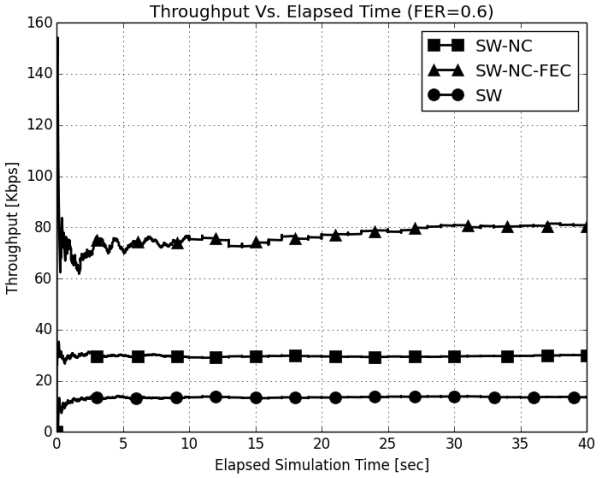

Fig. 4: Throughput of SW, SW-NC and SW-NC-FEC when FER=0.6

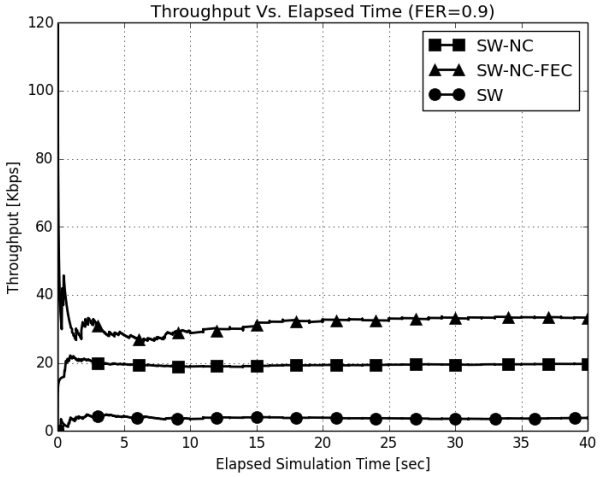

Fig. 5: Throughput of SW, SW-NC and SW-NC-FEC when FER=0.9

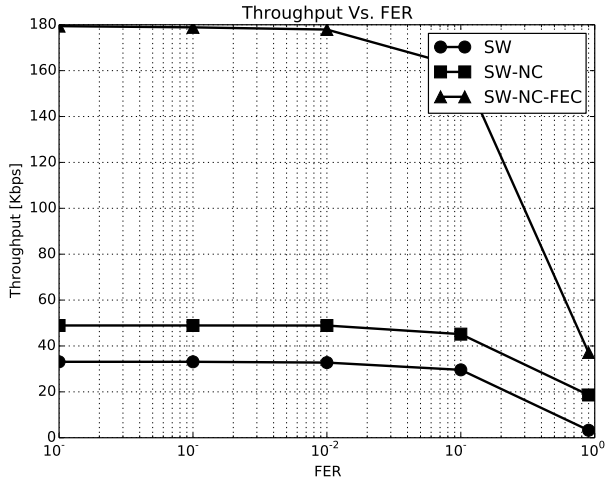

Fig. 6: Throughput of SW, SW-NC and SW-NC-FEC over a wide range of FER

of our proposed system. This is due to the fact that number of re-transmissions in SW-NC-FEC approaches the number of re-transmissions of the SW-NC as FER approaches 1. As such, the larger the error rate the more packets SW-NC-FEC needs to transfer and the closer its behavior is to SW-NC in terms of number of packets sent. Thus the cost of packet loss is higher when FEC is added.

In addition, SW-NC-FEC is greatly affected by packet generation rate. This can be seen in Figure 7, where SW-NC-FEC steeply climbs as packet generation rate increases. Conversely, SW-NC-FEC does not perform well with small generation rate. In fact, it performs worse than SW-NC when generation rate is below 0.03 packets/s.

Finally, scalability in terms of increase in number of packets encoded is investigated (see: Figure 8). It is found that SW-NC-FEC's performance increases linearly with increase in number of packets while SW-NC's performance decreases linearly (although slightly) with increase in number of packets.

B. Application

Table II summarizes the characteristics of SW, SW-NC and SW-NC-FEC. Base on this comparison, it can be seen that SW-NC shows more stable performance in a noisy environment and preforms better in a very low traffic network. Furthermore, the system is less complex than SW-NC-FEC, as such it shorter code and less processing overhead. Therefore, SW-NC is better for low traffic network systems where low processing power devices are used like wireless sensor networks, while SW-NC-FEC is best for networks that require large amount of data to be transmitted over a low noise channel.

TABLE II: Comparison between SW, SW-NC and SW-NC-FEC

	SW	SW-NC	SW-NC-FEC
Complexity	Simple	More complex	Most complex
Memory overhead	Less	More	Most
Throughput @ FER=0	33.0kbps	48.4kbps (147.0% of SW)	180.0kbps(545.5% of SW)
Throughput @ FER=0.9	6.0kbps	18.0kbps (300% of SW)	36.0kbps(600% of SW)
Decay in throughput with increase in FER	Faster	Fast	Fastest
Throughput as No. of packets encoded increase	-	Falls linearly	Increases linearly

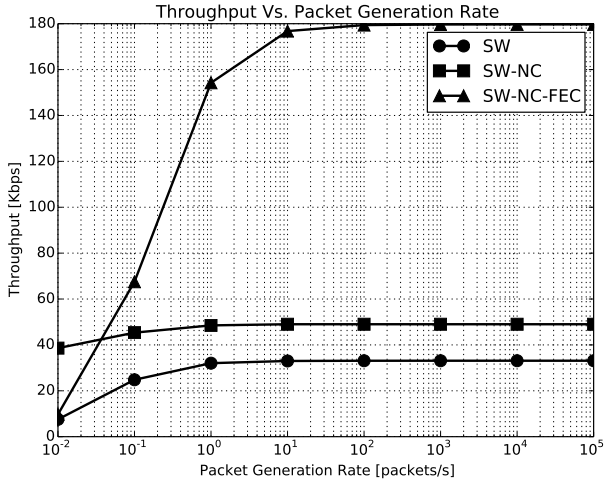

Fig. 7: Throughput of SW, SW-NC and SW-NC-FEC against packet generation rate

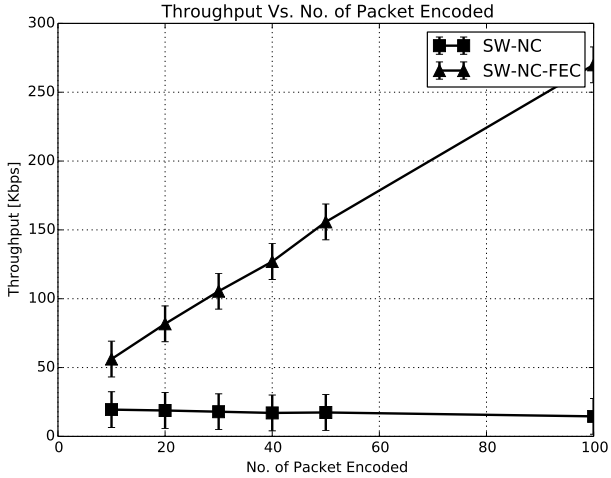

Fig. 8: Throughput of SW-NC and SW-NC-FEC against number of packets encoded

VI. Conclusions and Future Work

In this paper, the throughput of a SW-ARQ using Vandermonde matrix for network coding is analyzed. The study shows that NC with FEC is highly profitable and that adding FEC to the system in [8] improves its throughput. Furthermore, the study show that SW-NC-FEC has higher throughput than SW-NC, but its throughput falls more sharply with increase in frame error rate. Conversely, it is also found that SW-NC-FEC rises more sharply when packet generation rate increase. Finally we found that Forward Error Correction helps SW-NC-FEC to produce more throughput as Vandermonde matrix is enlarged, while SW-NC degrades.

Due to the fact that the connection between the transmitters and the receivers is many-to-one, there is a possibility of performance degradation when the number of transmitters is increased. As their number increases, a bottle neck is may formed at the encoder and this may cause delay in the network. However, the degree of decay in performance of a network-coded network has not been investigated. It is important to ascertain when the throughput starts to degrade and how bad is the degradation, and this forms the basis of our future work.

Acknowledgements

We would like to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for its continuous support to academic research activities.

References

[1] Y. Qin and L.-L. Yang, "Steady-state throughput analysis of network coding nodes employing stop-and-wait automatic repeat request," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1402–1411, Oct 2012.

[2] M. Zhang, "Major automatic repeat request protocols generalization and future develop direction," in *6th International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII)*, vol. 2. IEEE, 2013, pp. 5–8.

[3] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[4] J.-Y. Lee, W.-J. Kim, J.-Y. Baek, and Y.-J. Suh, "A wireless network coding scheme with forward error correction code in wireless mesh networks," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, 2009, pp. 1–6.

[5] A. A. Bruen and M. A. Forcinito, *Cryptography, information theory, and error-correction: a handbook for the 21st century*. John Wiley & Sons, 2011, vol. 68.

[6] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network coding: An instant primer," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, Jan. 2006. [Online]. Available: http://doi.acm.org/10.1145/1111322.1111337

[7] S. Luyi, F. Jinyi, and Y. Xiaohua, "Forward error correction," in *Fourth International Conference on Computational and Information Sciences (ICCIS)*. IEEE, 2012, pp. 37–40.

[8] A. Alsebae, M. Leeson, and R. Green, "The throughput benefits of network coding for sw-arq communication," in *27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, March 2013, pp. 854–859.

[9] S. De Vuyst, S. Wittevrongel, and H. Bruneel, "Delay analysis of the stop-and-wait arq protocol over a correlated error channel," in *Proc. of the second international working conference on performance modelling and evaluation of heterogeneous networks HET-NETs' 04*, 2004, p. 21.

[10] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell system technical journal*, vol. 39, no. 5, pp. 1253–1265, 1960.

[11] Q.-T. Quoc-Tuan Vien, L.-N. Tran, and H. X. Nguyen, "Network coding-based arq retransmission strategies for two-way wireless relay networks," in *Software, Telecommunications and Computer Networks (SoftCOM), 2010 International Conference on*. IEEE, 2010, pp. 180–184.

[12] A. Antonopoulos and C. Verikoukis, "Network coding based cooperative arq scheme," *arXiv preprint arXiv:1201.4650*, 2012.

[13] B. Li and D. Niu, "Random network coding in peer-to-peer networks: from theory to practice," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 513–523, 2011.

[14] Y. Liu, Z. Feng, and P. Zhang, "A novel arq scheme based on network coding theory in cognitive radio networks," in *IEEE International Conference on Wireless Information Technology and Systems (ICWITS)*. IEEE, 2010, pp. 1–4.

[15] Y. Qin and L.-L. Yang, "Throughput analysis of stop-and-wait automatic repeat request scheme for network coding nodes," in *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*. IEEE, 2010, pp. 1–5.

[16] X. Chen and D. Leith, "Frames in outdoor 802.11 wlans provide a hybrid binary-symmetric/packet-erasure channel," *arXiv preprint arXiv:1209.4504*, 2012.

[17] N. Ilievska and D. Gligoroski, "Error-detecting code using linear quasigroups," in *ICT Innovations 2014*, ser. Advances in Intelligent Systems and Computing, A. M. Bogdanova and D. Gjorgjevikj, Eds. Springer International Publishing, 2015, vol. 311, pp. 309–318. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-09879-1_31

[18] A. Al-Shaikhi and J. Ilow, "Vandermonde matrix packet-level fec for joint recovery from errors and packet loss," in *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, Sept 2008, pp. 1–6.

[19] G. S. Fishman, *Principles of discrete event simulation.[Book review]*. John Wiley and Sons, New York, NY, 1978.

[20] D. P. Kroese, T. Taimre, and Z. I. Botev, *Discrete Event Simulation*. Wiley Online Library, 2011.

[21] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science and engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[22] S. Vaingast, "The environment," in *Beginning Python Visualization*. Apress, 2014, pp. 31–53. [Online]. Available: http://dx.doi.org/10.1007/978-1-4842-0052-0_2