TechRxiv
Powered by IEEE

Novel One Item Network Coding Vectors

This paper was downloaded from TechRxiv (https://www.techrxiv.org).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

10-06-2021 / 12-06-2021

CITATION

AbuDaqa, Anas; Mahm, Ashraf (2021): Novel One Item Network Coding Vectors. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.14766396.v1

DOI

10.36227/techrxiv.14766396.v1

Novel One Item Network Coding Vectors

Anas A. Abudaqa, *Member, IEEE,* and Ashraf S. Mahmoud *Member, IEEE,*

Abstract

Herein the problem of network coding coefficients overhead is revisited. A novel approach, based on modular arithmetic and prime numbers, and is influenced by the Chinese remainder theorem (CRT), is proposed to reduce the coefficients overhead by augmenting only one item coefficient of maximum size of four bytes to the data packet. The proposed approach successfully addresses all the limitations of the previous methods including the limitations on the generation size and density of the packets in the generation, rocoding on the intermediate nodes, and creating innovative coding vectors. Probabilistic theoretical analysis and experimental work show a superior performance of the proposed approach in terms of coefficients overhead ratio, download time, throughput, and packets drop rate. This evaluation has considered three types of networks: wireless sensors network for Internet of things, conventional wireline Ethernet, and Peer-to-Peer content distribution.

Index Terms

network coding, network coding overhead, coefficients vector, Chinese remainder theorem , RLNC.

I. Introduction

With Random Linear Network Coding (RLNC) [1], the coded packet of a generation of size n is a linear combination of n plain packets such that the coded packet contains the original data

A. Abudaqa and A. Mahmoud are with the Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, emails: anas86@ieee.org, ashraf@kfupm.edu.sa).

plus a vector of coefficients (ζ), a.k.a coding vector, as a header. The size of the coding vector depends on the Galois Field (GF) size, the generation size, and the coding vector representation. On the receiver side, it is needed to receive the $n \times n$ coefficients matrix, a.k.a the transfer matrix, along with n coded packets to solve a system like in (1) and recover the original packets. The total coefficients overhead of RLNC is $n \log_2 q$ and $n^2 \log_2 q$ per packet and per generation, respectively, where q is the GF size.

$$\begin{bmatrix} P_{c1} \\ P_{c2} \\ \vdots \\ P_{cn} \end{bmatrix} = \underbrace{\begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,n} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ e_{n,1} & e_{n,2} & \cdots & e_{n,n} \end{bmatrix}}_{\text{n*n coefficients matrix}} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \tag{1}$$

Although the coefficients overhead can be considered small compared to large data packets, there are some cases in which this overhead can be significant and consumes noticeable network bandwidth and power of the devices. For instance, in wireless sensors networks (WSN) where packets are short in length and devices have limited power, the coefficients overhead can quickly become prohibitive [2]. Even in wired networks if only standard maximum transmission units (MTU), such as 1500 bytes are permitted, the header fraction can be too high [3]. Moreover, in Peer-to-Peer (P2P) content distribution networks where huge sized files are shared, e.g., 10 Gigabytes (GB) or larger, the overall coefficients overhead incurred is in hundreds of Megabytes (MB). The following three examples demonstrate cases where the coefficients overhead is significant.

Example 1 ([4]): Consider a wireless sensors network that contains multi-sources which want to send 50 packets to a sink. If network coding is applied over $GF(2^8)$, then the coding vector is 50 F_{256}^{50} symbols which translates to 50 bytes. But in a such network the overall packet size is about 100 Bytes, e.g. zigbee packet, which means that the overhead ratio is 50%.

Example 2: Consider a wired network that uses $MTU = 1500$ Bytes as a packet size. In such network the generation size is typically large. Hence if we assume $n = 512$ and the coefficients

are drawn from $GF(2^8)$, then the coding vector is 512 bytes which is more than one third of the MTU.

Example 3: Consider a P2P content distribution network that uses 16KB as a packet size and shares a 10GB File. Usually in such scenario the best generation size is 20MB [5], then the file is segmented to $10GB/20MB = 512$ generations. Each block in the generation is 16KB, then number of blocks in a generation is $20MB/16KB = 1280$ blocks. Thus, the coefficients matrix is 1280×1280 per generation and if the coefficient are drawn from $GF(2^8)$ and each represented as 1 byte, then the total coefficients overhead of the file is $1280B \times 1280B \times 512 =$ 800MB per the file.

A. Methods in Literature

Many approaches in the literature are proposed to address the problem of network coding coefficients overhead. Most of these approaches are based on using sparse coding vectors, i.e., each coding vector contains only small numbers of nonzero coefficients, and thus the coding vectors can be compressed.

Mahdi J. et. al. in [3] propose first compressing coefficients vectors scheme which is based on error correcting codes and using parity check matrices. The scheme considers m nonzero coefficients, where $m << n$. The scheme results in coefficients overhead equals to $2m$ per packet.

S. Li et. al. in [6] improve on the previous scheme by utilizing a method of erasure codes and adding a segment ID to the header to enumerate the IDs of the m nonzero coefficients. The overall compression rate is improved and coefficients overhead becomes $m + n/\log q$. Further, the coefficients overhead becomes $m + (\log_2 n)/\log_2 q$ when a list decoding [7] method is used. However to achieve the latter rate a GF with a large size and more complicated decoding algorithms are required. With ordinary GF and efficient decoding algorithms the coefficients

overhead becomes close to:

$$m - \frac{m^2}{n} + \frac{(\log_2 n)}{\log_2 q}. \tag{2}$$

Danilo.G et. al. in [8] propose another sparse method that only allows selecting coefficients from a small set, Q, of $GF(q)$ elements. Specifically, Q is a subset of the primitive elements of $GF(q)$, and can be agreed upon among the network nodes in advance. Thus, the sender instead of sending the m coefficients values as an n- coding vector, it can send only a bit vector that contains the coefficients indexes in Q and their positions in the coding vector. The overall coefficients overhead of this scheme is $m(log_2|Q| + log_2 n)$ bits.

Ye.L et. al. in [9], propose a method that splits the coefficients overhead into two vectors. The first one contains the m coefficients values. The second vector is an n-bit vector of 0's and 1's. on the receiver, the 1's are replaced with the m coefficients. The overall coefficients overhead of this scheme is

$$m \log(q)/8 + \lceil n/8 \rceil \, bytes. \tag{3}$$

A scheme that is very related to our work is proposed in [10]. The scheme reduces the coefficients overhead per packet to be one symbol based on Vandermonde matrices. In the Vadermonde matrix, the entire row can be generated using a single seed. However, this scheme has two apparent limitations. First, because of the cyclic characteristic of the finite field, the generation size is restricted to be $n <= GF(q)$ which means that in practical scenarios the generation size cannot exceed 32 packets. Second, recoding on intermediate nodes is highly likely to generate linearly dependent coding vectors. Another scheme, which uses a single seed that is transferred from the sender to the receiver, is proposed in [11]. However, this method requires synchronization among the participant nodes and it completely lacks the recodability on the intermediate nodes.

Niloofar.Y et. al. in [12] propose revolve codes (ReC) which are based on drawing a common element from a large GF, namely $q \geq 2^{16}$ to guarantee low probability of linear dependency,

and another n elements from $GF(2^2)$. The small elements are interpreted as one item t and sent along with the large element as a compressed coefficients vector. The receiver generates the complete coefficients vector by repeatedly shifting the small elements item and each time adding to the large element. The overall coefficients overhead of this method is ($\log_2 q + n * t$).

Most of the above studies [3], [8], [10]–[12] are unreliable in terms of recoding, generating linearly independent vectors, or\and scalability, namely, they do not support wide range of generation sizes. Moreover, the most reliable methods of these studies [6], [9] are sparse, which usually requires more complex recoding strategies to retain the structure of the sparse code [9], [12].

While this paper proposes a novel method that is fully dense, the proposed method incurs smaller overhead and performs more reliably compared the sparse coding methods for the most cases. Also, we believe that it is the most energy saver for small, e.g., sensors, devices.

B. Contributions and The Paper Organization

In this paper we address the previously mentioned problems and provide the following contributions:

- We propose a new method, based on modular arithmetic and prime numbers referred to as the One Item Network Coding Vectors (OINCV). The method efficiently compresses the coefficients vectors to a tiny constant value that often does not exceed 4 bytes.
- The proposed method is dense, recodable, and reliable in the sense that it can be real alternative to RLNC without any limitations.
- The proposed method is scalable and it can be used over a wide range of applications and devices, i.e., from wireless sensors networks to high performance computers.
- Both theoretical analysis and simulation are provided to prove the correctness and features of the proposed method.
- Theoretically, the proposed method is compared with all state of art studies in terms of

coefficients overhead ratio, and encoding cost.

– Experimentally, the proposed method is compared with all reliable studies in terms of coefficients overhead ratio, download time, throughput, and packets drop rate. This evaluation has considered three types of networks: wireless sensors network for Internet of things, conventional wireline Ethernet, and P2P content distribution.

The rest of the paper is organized as follow. Section II presents important preliminaries and the design model. The theoretical analysis and the proof of correctness is detailed in Section III, while the experimental work and results are presented in section IV. Section V provides the conclusion and outlines the future work.

II. Preliminaries and Design Model

Let P be a prime and $Z_P = \{0, 1, ..., P-1\}$, if addition and multiplication are defined in Z_P as being modulo P, Z_P can be defined as the prime GF of order P, $GF(P)$. On the other hand, $GF(2^r)$ or $GF(q)$ where $q = 2^r$ and $r >= 2$ is referred to as the extended GF. However, elements in the extended GF are presented as polynomials of degree $r-1$ and arithmetic operations are defined as being modulo irreducible primitive polynomial $P(X)$ of degree r; such primitive polynomials are analogous to the prime numbers in Z.

Herein, both prime and extended GF are used. $GF(2^r)$ and $GF(q)$ are used interchangeably and when $GF(P)$ and $GF(q)$ are mentioned together, it means that P is always the largest prime number less than q. Also, P is always notated by big P, while little p_i notation is used to refer to the other primes in $GF(P)$.

The Chinese remainder theorem (CRT) is used to solve a set of congruent equations with one variable but different moduli, which are relatively prime.

Theorem 1 (The Chinese remainder theorem (CRT)): Let p_1, p_2, ..., p_n are relatively prime

positive integers, then the following equations have a unique solution modulo $M = p_1 * p_2 * ... * p_n$.

$$X \equiv a_1 \pmod{p_1}$$

$$X \equiv a_2 \pmod{p_2}$$

$$\vdots$$

$$X \equiv a_3 \pmod{p_n}$$

The proof and details about this theorem can be found in [13], [14].

Definition 1 (Code Density [15]:): It refers to the ratio of non-zero elements in an coefficients vector relative to the entire size of the vector. It is calculated as:

$$d(c) = \frac{\sum_{i=1}^{n} e_i \neq 0}{n} = \frac{m}{n} \tag{4}$$

Full density can be achieved by choosing a coefficient for each cell in the coefficients vector based on random uniform distribution. On the other hand, sparse density can be achieved by allowing several zeros to be positioned in the coefficients vector. Full density is desirable as it yields high degree of innovation amongst packets, while it results in higher coefficients overhead. In practice, sparsity is widely used as it allows the coefficients vector to be compressed and hence yields reduced coefficients overhead. However, the higher the sparsity, i.e., code density becomes closer to 0, the higher the risk to get linearly dependent vectors which leads to even worse overhead. Herein, we set the code density to 0.4, i.e., $m = n/2.5$.

In what follows, we propose a dense coding vectors technique referred to as OINCV based on modular arithmetic and prime numbers. It incurs only 2, 3 or at most 4 bytes as an overhead per packet.

A. *Primarily step : generating the primes*

For a generation of size n, n primes are generated independently in each network node such that the n generated primes are identical in all the nodes.

Definition 2 (Generation Moduli Set (GMS)): For a network coding generation of size n that runs GF operations, e.g. addition and multiplication, over $GF(2^r)$, the generation moduli set is defined as the set of the largest consecutive $n-1$ primes picked up from $GF(P)$ besides P itself, such that GMS=$\{p_1$,p_2, ... p_{n-1},$P\}$.

For example and without loss of generality the GMSs for a 4-packets generation that operates over different GFs are listed in table I.

TABLE I: Examples of GMSs

Extended GF	Prime GF	GMS
$GF(64)$	$GF(61)$	{49, 53,59,61}
$GF(128)$	$GF(127)$	{107,109,113,127}
$GF(256)$	$GF(251)$	{233,239,241,251}

The reasons behind selecting the largest n primes are twofold: 1) to make the nodes synchronized, namely all the nodes generate the same GMS without the need to transfer the primes and 2) to leverage the items of the coding vector, namely, the modulo primes operation will give more items combination within a coding vector, thereby reducing linearly dependent vectors probability. We emphasize that $GF(P)$ is used to create the GMS, and generate or regenerate the coefficients vectors, while the needed operations by network coding are still performed over $GF(q)$.

B. Encoding

Definition 3 (Traversed Coefficient (I)): is an integer number selected randomly by the sender such that it is a 2, 3, or 4 bytes and then it is transmitted as a header of the coded packet to generate the entire coefficients vector on the receiver side.

Definition 4 (Allowed Selection Range (ASR)): is a set where the traversed coefficients are

Fig. 1: Two network coding packets: (a) RLNC's packet, and (b) OINCV's packet

allowed to be selected. e.g., the set of integer numbers in the range $[P+1, 2^{16}-1]$, $[P+1, 2^{24}-1]$, or $[P+1, 2^{32}-1]$. An appropriate ASR is chosen based on the generation size

For instance, for an RLNC with generation size of 32, it can use $GF(P = 251)$ to have a GMS with 32 primes and the RLNC operations can be run over $GF(q = 256)$, and hence, the appropriate ASR for such system is in range $[252, 2^{16} - 1]$.

Encoding begins by generating a proper GMS that matches the generation size, thereafter, the sender selects a traversed coefficient I from the ASR set, and then a vector of integers is constructed as follows:

$$[I \bmod p_1, I \bmod p_2, ..., I \bmod p_{n-1}, I \bmod P]$$

These vector elements are $GF(P)$ integers, therefore, they can easily be transformed to the corresponding $GF(q)$ elements, such that the coding coefficients vector becomes as:

$$\zeta = [e_1, e_2, ..., e_{n-1}, e_n]$$

To compose the coded packet, each coefficient from ζ is multiplied by its corresponding plain packet and then packets are combined, exactly as in RLNC. However, instead of sending the complete coding coefficients vector ζ, the sender sends only the traversed coefficient I along with the payload. For example assume a generation size of 32 packets and RLNC is applied over $GF(2^8)$. Fig. 1 shows how OINCV can reduce the network coding overhead from 32 bytes to only 2 bytes.

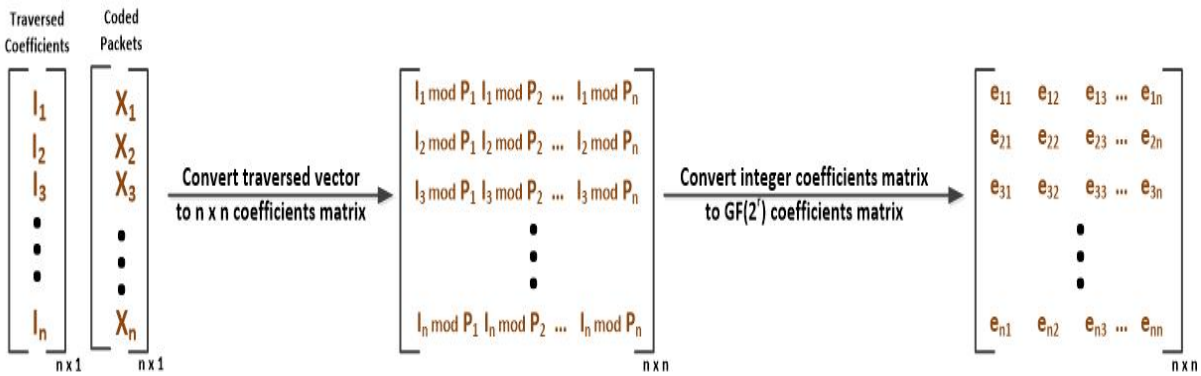


Fig. 2: Decoding process on the receiver side

C. Decoding

Once the receiver received the coded packet, it uses the traversed coefficient to construct the coefficients vector ζ, in the same way the sender does. The receiver keeps filling the coefficients matrix until it is a full rank, and then solves using Gaussian elimination to get the original generation packets. The steps are presented in Fig. 2.

D. Recoding

An intermediate node, e.g. a router, recodes the incoming packets by randomly selecting two packets out of all the received packets and new two coding coefficients ($\hat{e_1}$ and $\hat{e_2}$) from $GF(2^8)$. The intermediate node then multiplies each coefficient by its corresponding packet's payload. Thereafter, linearly combines the new payloads together to produce the outgoing packet. Two packets are sufficient since the proposed method is fully dense. On the other hand, the overhead part is composed by attaching the new selected coefficients ($\hat{e_1}$, $\hat{e_2}$), along with the original traversed coefficients (I_1, I_2) of the two selected packets to the outgoing packet as shown in Fig. 3 (a). The reason for attaching the coefficient as opposed to combining them is that if ($\hat{e_1}$, $\hat{e_2}$) are multiplied by (I_1, I_2), then (I_1, I_2) are almost impossible to be recovered on the receiver side. Thus, attaching these two coefficients reduces the extra overhead from n to a maximum

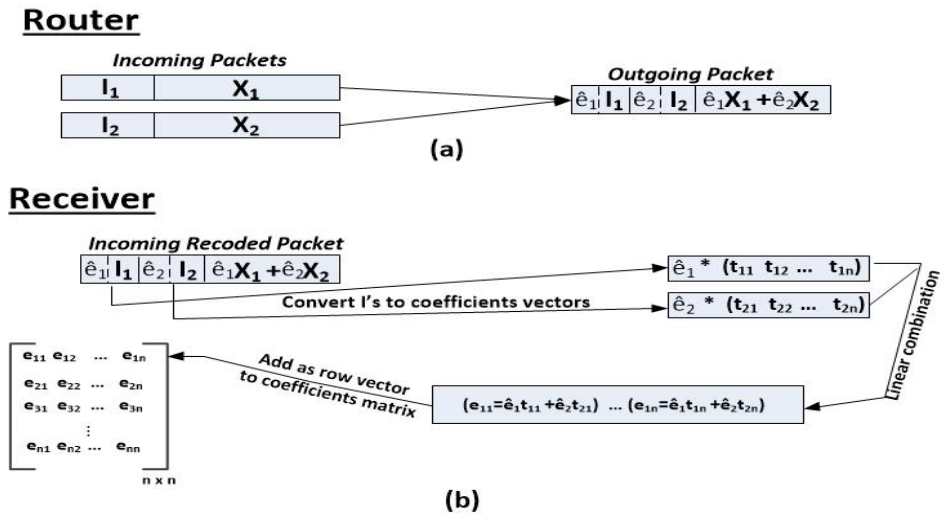


Fig. 3: (a) Recoding process on an intermediate node and (b) Decoding of a recoded packet on the receiver.

10 bytes as will be shown shortly.

When the receiver receives the recoded packet, it uses the traversed coefficients to construct the coding coefficients vectors as shown in the previous subsection. Next, each attached coding coefficient ($\hat{e}_i$) is multiplied by its corresponding coefficients vector ((I_1). Next, the two coding vectors are linearly combined to produce one vector. Finally, this vector is added as a row vector to the coefficients matrix. The complete process is illustrated in Fig .3 (b). The payload part of the recoded packet is not touched on the receiver side and it is added as is to the coded packets column.

Two essential questions about the proposed method arise. First, since the servers are not co-located and have no coordination, and the traversed coefficients are selected randomly, then selecting the same traversed coefficient by different two servers leads to constructing two identical coefficients vectors and as a consequence two linearly dependent vectors. Hence what is the probability that two or more servers select the same traversed coefficient? The second question

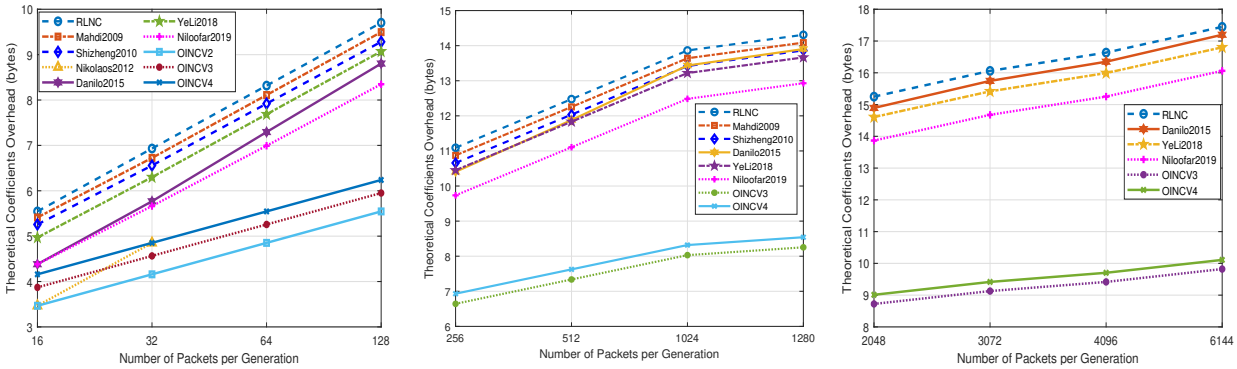
is will two different traversed coefficients, I_1 and I_2, generate the same coefficients vectors after the modulo operation over the GMS? These questions are addressed and analyzed in the next section.

III. Theoretical Analysis and Proof of Correctness

In this section, we provide theoretical analysis and prove the correctness of the proposed method. In addition, we compare the proposed method with all the previous studies in terms of coefficients overhead and encoding cost.

A. Analysis of network coding coefficients overhead

If recoding is not required, then the coding coefficients overhead incurred by OINCV is a very tiny constant per packet regardless of the generation size, namely, it is 2, 3, or 4 bytes per packet and $2n$, $3n$, $4n$ bytes per generation. In case of recoding, the analysis considers that a recoded packet is composed by combining two coded packets only and the receiver is one-hop away from the server, i.e, only one intermediate node is assumed between the sender and the receiver. Thus in the recoding case, the overhead contains two traversed coefficients plus two selected coefficients ($\hat{e}_1, \hat{e}_2$), on the intermediate node, whose size is 2 bytes and hence the overall coefficients overhead incurred by the proposed method per packet becomes 6, 8, or 10 bytes. Whether recoding is required or not, the proposed method compared to the other state of art studies can reduce the coefficients overhead to a very large extent. Recall that $m = \lceil n/2.5 \rceil$, Fig. 4 (a), Fig. 4 (b), and Fig. 4 (c) present results for theoretical coefficients overhead per generation (plotted on log scale), when the recoding is not required, for small, moderate, and large generations sizes, respectively. The evaluation is performed for OINCV using 2, 3, and 4 bytes traversed coefficient, referred to as OINCV2, OINCV3, and OINCV4, respectively. The figure shows that OINCV always incurs the minimum overhead and the difference in the overhead value between OINCV and the other methods increases as the generations sizes get larger. Moreover,



((a)) Small generations ((b)) Moderate generations ((c)) Large generations

Fig. 4: Log scale plot of network coding coefficients overhead without recoding

the figure shows that the proposed method is viable for all the generations sizes, while some methods collapse for moderate and large sizes.

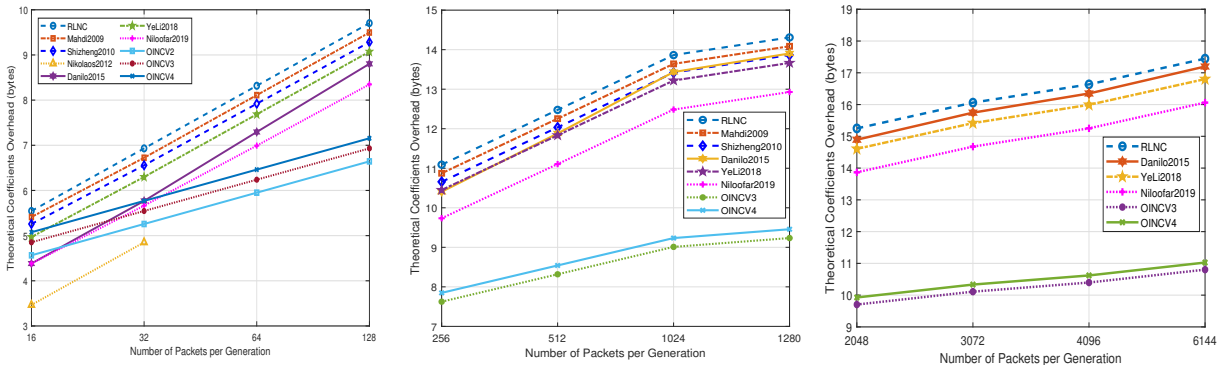
Figure. 5 shows the same results but with recoding. The same patterns are observed as for the case without recoding, with the exception for the generation size of 16. The method proposed in [10] has less coefficient overhead when the generation size is 16, but it is prone to high probability of linear dependency amongst coded packets if recoding is required. In addition, the method is viable only for $n \leq 32$ [10].

B. Analysis of selecting different values from ASR

To decode a generation of n packets successfully, OINCV needs n distinguishable traversed coefficients to be selected from ASR. we examine the probability of selecting n distinguishable traversed coefficients.

Proposition 1: The probability that n traversed coefficients selected from an ASR are distinguishable (P_{dist}) is given by:

$$P_{dist}(I_1, I_2, ..., I_n) = \prod_{i=1}^{n} \frac{|ASR| - (i-1)}{|ASR|}. \tag{5}$$



((a)) Small generations ((b)) Moderate generations ((c)) Large generations

Fig. 5: Log scale plot of network coding coefficients overhead with recoding

Proof: For the first selection, the probability that the traversed coefficient, I_1, is distinguishable is equal to 1, $P_{dist}(I_1) = 1$. For the second selection, $P_{dist}(I_2) = \frac{|ASR|-1}{|ASR|}$, for the third selection, $P_{dist}(I_3) = \frac{|ASR|-2}{|ASR|}$ and so on for the next selections. Finally, for the n-th selection , $P_{dist}(I_n) = \frac{|ASR|-(n-1)}{|ASR|}$. Since these are independent events, then the overall probability of selecting distinguishable traversed coefficient from the ASR for a generation of size n is given by:

$$P_{dist}(I_1) \times P_{dist}(I_2) \times ... \times P_{dist}(I_n) = \prod_{i=1}^{n} \frac{|ASR| - (i-1)}{|ASR|}$$

■

Figure. 6 shows the evaluation of probability of selecting distinguishable traversed coefficients based on (5) and also using simulation for different generation sizes. As before, traversed coefficients of sizes 2 ,3, and 4 bytes are considered and denoted in the figure as *OINCV2*, *OINCV3*, and *OINCV4*, respectively. For the simulation, each experiment is run for 1000 repetitions then the average result is presented. From the figure it can be concluded that the 2 bytes traversed coefficient can be only used with small generations sizes, i.e. size ≤ 32, since GMS for these generations can be picked up from $GF(P = 251)$ and the network coding operations can be

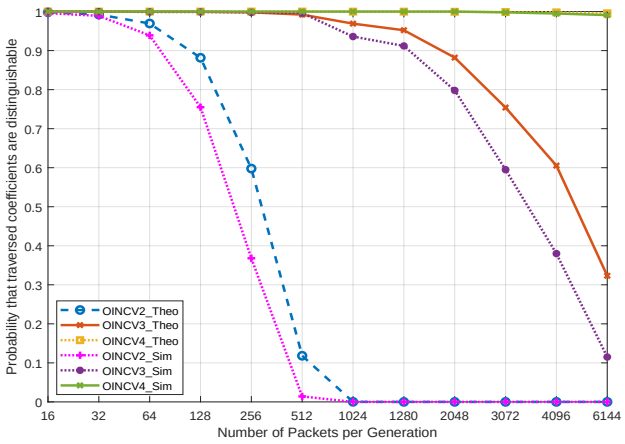


Fig. 6: Probability that traversed Coefficients are distinguishable; theoretical vs simulation

run over $GF(2^8)$. Larger generations, with size ≥ 64, must not consider the 2 bytes traversed coefficient for the following reasons: First, they need that their network coding operations run over a GF greater than $GF(2^8)$, for example, there is no 64 primes less than 256 and thus a 64 packets generation needs to use $GF(P = 509)$ for generating GMS elements and $GF(2^9)$ for network coding operations which leads to chunk a large packet into a 9 bits blocks which in turn violates the computer arithmetic and network convenience that stores and sends the data as multiple of 8-bit words. The second reason is that the probability of getting distinguishable traversed coefficients is decreased as the generations become larger than 32 as shown in the figure. When the traversed coefficient is of size 3 or 4 bytes, the operations coefficients are generated over $GF(2^{16})$ since the GMS is picked up from $GF(P = 65521)$ and the figure shows that the 3 bytes traversed coefficient can be used with moderate generations, i.e. between 64 and 512, and the 4 bytes traversed coefficient can be used with large generations, indeed, 4 bytes traversed coefficient can be used for any generation at the cost of extra one or two bytes overhead.

Theorem 2: The arbitrarily selected n traversed coefficients from an appropriate ASR are

distinguishable with high probability.

Proof: The proof is constructive by substituting in (5) and the observations from Fig.6.

- for small generations that contain 32 packets or less, traversed coefficients $(I_1, I_2, ..., I_n)$ can be selected from an ASR whose range and size are $[252, 2^{16} - 1]$ and $|ASR| = 65284$. Consequently, $P_{dist}(I_1, I_2, ..., I_n) \geq 0.99$.
- for moderate generations that contain packets between 64 and 512, traversed coefficients $(I_1, I_2, ..., I_n)$ can be selected from an ASR whose range and size are $[65522, 2^{24} - 1]$ and $|ASR| = 8323086$. Consequently, $P_{dist}(I_1, I_2, ..., I_n) \geq 0.99$.
- for large generations that contain packets between 1024 and 6144, traversed coefficients $(I_1, I_2, ..., I_n)$ can be selected from an ASR whose range and size are $[65522, 2^{32} - 1]$ and $|ASR| = 4294901774$. Consequently, $P_{dist}(I_1, I_2, ..., I_n) \geq 0.99$.
- Moreover, all these calculations are supported by simulation as shown in Fig. 6.

■

C. Analysis of generating two identical coding vectors

We showed in the previous subsection that the probability of selecting two identical traversed coefficients can be minimized to approach 0. In this subsection, we address the following two questions: (1) do two different traversed coefficients modulo the same GMS generate the same coding coefficients vector? (2) what is the probability that two or more different coding coefficients vectors are linearly dependent?

Theorem 3: Two different traversed coefficients modulo the same GMS do not generate identical coefficients vectors.

Proof: This can be proved by contradiction. Assume we have a coding coefficients vector $\zeta = [e_1 e_2 ... e_n]$ which is generated by a traversed coefficient (I_1) modulo GMS.

step1: Assume that there is another traversed coefficient (I_2) that can generate the same ζ.

step2: Let ζ vector be written as a system of simultaneous linear congruence:

$$I_1 \equiv e_1 \pmod{p_1}$$

$$I_1 \equiv e_2 \pmod{p_2}$$

$$\vdots$$

$$I_1 \equiv e_n \pmod{P}$$

since $p_1, p_2, ..., P$ are relatively prime, then utilizing the CRT the above system has a unique solution which is I_1. As a consequence, I_2 can not generate ζ unless $I_2 = (p_1 \times p_2 \times ... \times P) + I_1$ but since this value is far larger than I_1, it is out of ASR scope.

■

Finally and for the sake of completeness, we analyze the probability that two or more different coefficients vectors generated by the proposed method are linearly dependent.

It is well known that an $n \times n$ matrix over $GF(2^r)$ has full rank with high probability for large r, i.e., $r >= 8$ that is specified by [16]:

$$\prod_{i=0}^{n-1} \frac{q^n - q^i}{q^{n \times n}} \tag{6}$$

Since the coefficients vector for the proposed method is generated by the modulo operation over GMS, any of its coefficient's value cannot go beyond $P - 1$. Consequently, the $n \times n$ matrix values can only selected from $GF(2^r) - \{P, P+1, ..., 2^r\}$ group, whose size and element diversity are identical to $GF(P)$. This in turn reduces the selection diversity. However, for a P that is the largest prime smaller than 2^r, $GF(P)$ contains at least 98% of $GF(2^r)$ elements when $8 <= r <= 16$. Therefore an $n \times n$ matrix over $GF(P)$, is full rank with almost the same high probability.

Lemma 1: An $n \times n$ coefficients square matrix over $GF(2^r)$ where its elements are chosen

from $GF(P)$ is full rank with high probability that is given by:

$$\prod_{i=0}^{n-1} 1 - \frac{P^i}{P^n}. \tag{7}$$

Proof: Please refer to Appendix-A

■

D. Analysis of encoding cost

Definition 5 (Encoding Cost:): This refers to the time needed to compose an encoded packet from plain packets. Precisely, this time involves the addition and multiplication operations over $GF(2^r)$ needed to generate a such packet.

Since in the proposed method the GF operations are run over $GF(2^{16})$ for generations sizes $>= 64$, thus the whole data packet is represented by fewer GF symbols than the other methods which usually run GF operations over $GF(2^8)$ or $GF(2^4)$. This leads to minimizing the number of addition and multiplication operations. For instance, for a packet of size 1500 bytes if the used GF is $GF(2^8)$, then the packet is presented by 1500 symbols which means that an encoded packet needs $1500n$ addition and multiplication operations. On the other hand, if the used GF is $GF(2^{16})$, then the same packet is presented by 750 symbols which in turn minimizes the number of addition and multiplication operations to $750n$.

Although the cost for a single operation over $GF(2^{16})$ is more expensive than the cost of the same operation over $GF(2^8)$, the encoding cost of the proposed method is always less than that for RNLC and RLNC-like methods. This is because the number of operations is minimized to the half and most software and hardware implementations are using lookup tables for multiplication which makes the time per an operation is almost the same even when the field sizes are different. Table II presents results of the encoding cost measured in seconds for 128 packets, for generations range from 64 to 1280. The results show that whether the lookup tables, referred as LT, are used or not, OINCV has less encoding cost than RLNC.

TABLE II: Encoding cost (secs) of a 128 bytes packet

Generation Size	64	128	256	512	1024	1280
RLNC	0.133	0.212	0.717	0.754	1.533	1.910
OINCV	0.132	0.206	0.390	0.676	1.287	1.633
RLNC_{LT}	0.031	0.052	0.083	0.146	0.262	0.340
OINCV_{LT}	0.031	0.052	0.080	0.141	0.257	0.314

IV. Experimental Work

In this section, we consider a dissemination protocol that has multi servers which disseminate their data by performing network coding to one or multi receivers. The servers have no coordination among themselves. Also, we consider different file sizes that vary from 16 KB to 20 GB to cover wide range of applications and networks from wireless sensors to large scale P2P content distribution networks. OINCV is compared with the most reliable state of the art methods which are RLNC, the method of S.Li et. al. [6], and the method of Ye.L et. al. [9].

The evaluation considers the following performance metrics: coding coefficients overhead measured in KB, number of dropped packets on the receiver side, download time measured in terms of number of rounds, and throughput measured as number of packets per round. The remaining details needed for each network scenario are given in the respective subsections.

A. Wireless Sensors Network Case Study

In this case scenario, we consider the following settings:

– Data of sizes 16KB, 32KB ,64KB, and 128KB are shared by two servers to one client that is at zero or one hops away from the servers. The servers share their packets generation by generation employing a sequential scheduling policy. The client sends acknowledgement only when it receives all the generation's packets.

– We consider the Zigbee protocol [17] whose packet size is 128 bytes but since its header is 28 bytes, the payload is only 100 bytes in size.
– We assume a generation size of 32 packets [2] and a link bandwidth of 2 packets/round. Let the channel loss probability be $\epsilon = 0.2$ for data packets, while the loss probability for acknowledgements is $\alpha = 0.1$.

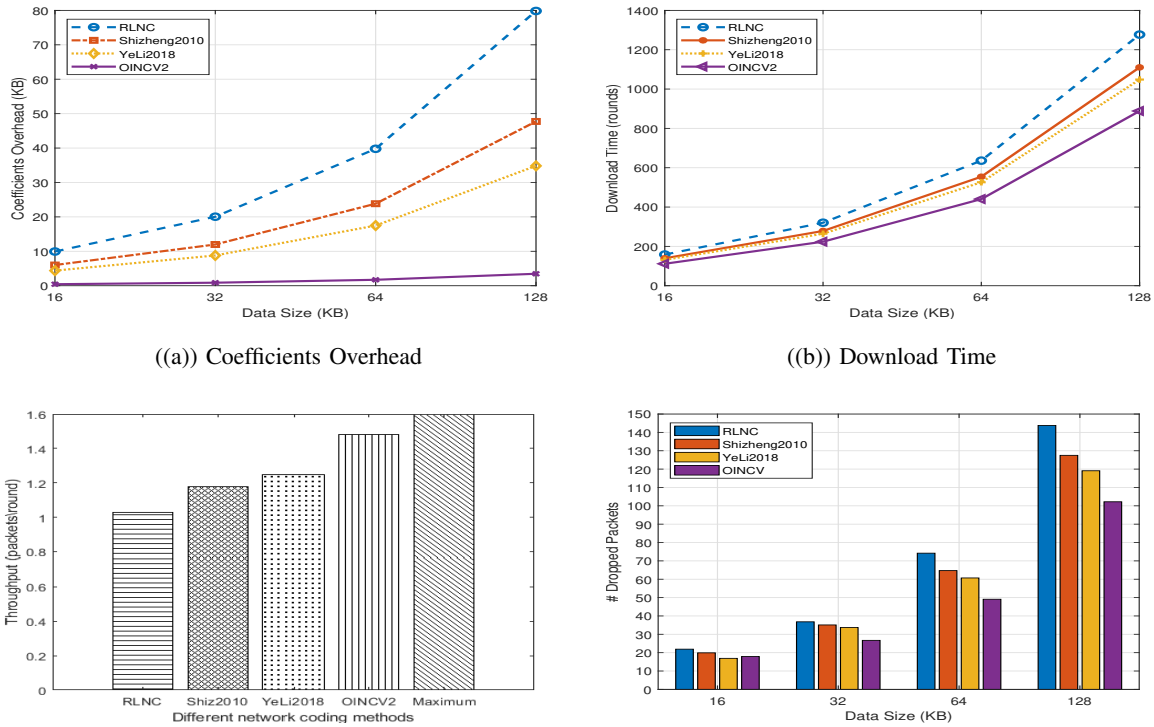
Based on these settings, we can specify the packet payload and coefficients overhead, and compute accordingly how many generations the data must be divided for each of the considered methods. For instance, when the data file size is 16KB and the RLNC method is used, then the coding coefficients overhead per packet is equal to the generation size, i.e., 32 bytes, and thus the remaining 68 bytes are for the payload. To specify the number of generations, first the data size is divided by the payload to get the number of packets, $\lceil 16KB/68B \rceil = 241$ packets, then the number of packets is divided by the generation size, $\lceil 241/32 \rceil = 8$ generations. For the methods proposed in [6] and [9], firstly the coefficients overhead is calculated according to (2) and (3), respectively, then the other specifications are determined. For OINCV the coefficients overhead is 2 bytes since GMS can be generated from $GF(251)$, and therefore I can be selected from $[252, 2^{16} - 1]$. Table III summarizes the specification for all methods. Figure. 7(a) shows results of the coefficients overhead in KB; the results indicate that the proposed method always incurs significantly less overhead that ranges from 0.43 KB to 3.5. While, the overhead for the next best

TABLE III: Specifications for different NC methods

NC Methods	Packet Payload (Bytes)	Packet Overhead (Bytes)	Number of generations			
			16KB	32KB	64KB	128KB
RLNC	68	32	8	16	31	61
Shizheng2010	78	22	7	14	27	53
YeLi2018	83	17	7	13	25	50
OINCV2	98	2	6	11	21	42

method ranges from 4.3 KB to about 35 KB. Figure. 7(b) shows the download time results in terms of rounds. Since the proposed method has the minimum overhead, then the payload of the packet contains more data for OINCV2 relative to other coding methods as shown in Table III. This means the data files is split to fewer packets and generations, and hence less download time. Figure. 7(c) shows the throughput results measured in packets/round. Although the bandwidth of the channel is 2 packets/around, the maximum (ideal) throughput is 1.6 packets/round since $\epsilon = 0.2$. The OINCV throughput is 1.48 packets/round which is the greater than that for the other methods and is very close to the ideal throughput. Lastly, Figure. 7(d) shows results of the number of dropped packets on the receiver side. Basically, packets are dropped on the receiver side when packets of a generation are received while the generation is already completed. This happens because of the loss of the acknowledgement or/and the absence of servers coordination. Since OINCV has always the minimum number of generations and thus fewer acknowledgement to be sent, it has always the minimum number of dropped packets.

Because the overhead for OINCV is constant and in contrast to the other methods, it is not affected by the generation size, it turns out that the generation size of 32 is not the optimal choice for OINCV. Indeed, OINCV can work using a generation size of 100 packets and perform better with the added cost of one extra byte of overhead for every packet. Figure. 8 compares OINCV of 32 packets for the generation size with OINCV of 100 packets for generation size, referred by $OINCV_{opt}$, in terms of coefficients overhead, download time, throughput, and number of dropped packets. Results show that the coefficients overhead of $OINCV_{opt}$ is slightly larger than OINCV and this is because for a generation size of 100 packets, 100 primes are required. Thus, to maintain the compatibility with computer arithmetic, GMS can be generated from $GF(P = 65521)$, I is selected from $[65522, 2^{24} - 1]$, and network coding operations run over $GF(2^{16})$. The results also show that all the other metrics are improved, e.g., throughput reaches 1.54 packets/round and the number of dropped packets is reduced by at least 60%.



((a)) Coefficients Overhead

((b)) Download Time

((c)) Throughput

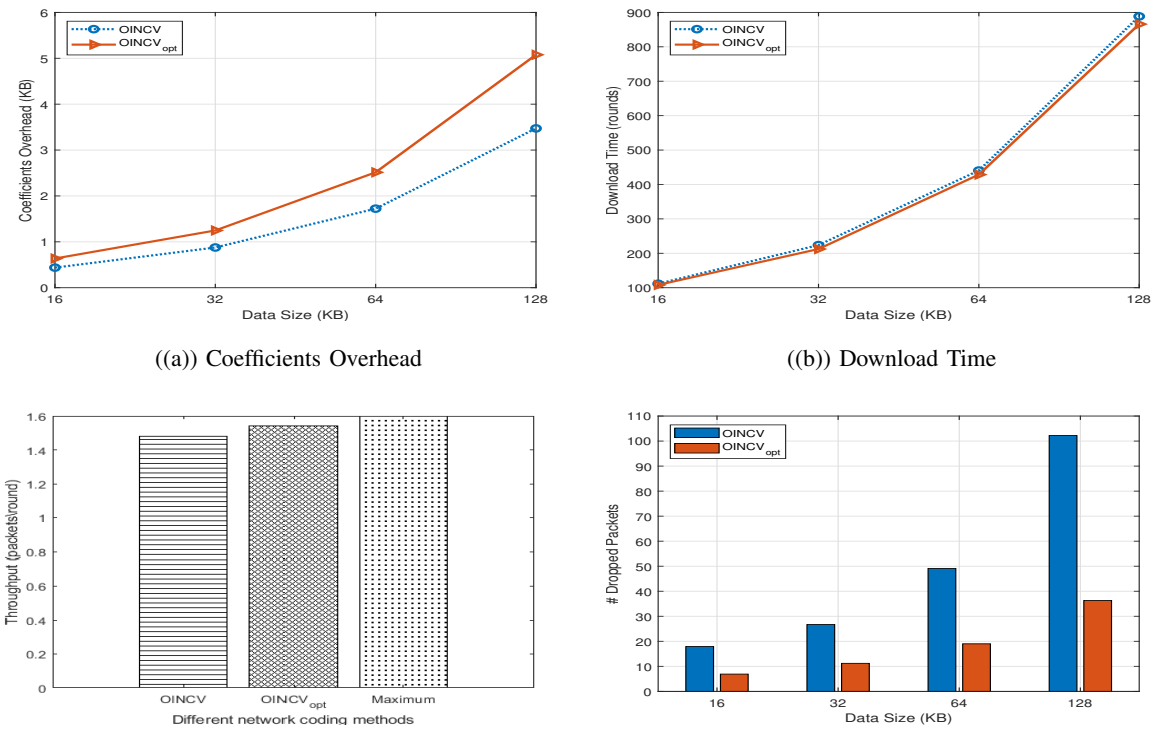
((d)) Dropped Packets

Fig. 7: Performance metrics of 0-hops WSN for different NC methods

B. Recoding

In this subsection, we repeat the previous experiments with the same settings except now an access point (AP) exists between the servers and the receiver, and thus recoding is required. We assume that the links between the servers and AP, and the one between the AP and the receiver have the loss probability of $\epsilon = 0.2$. Table IV summarizes the new specifications for OINCV and $OINCV_{opt}$, considering that recoding of the other methods does not need any extra cost.

Figure. 9 presents results of the new network topology for the same metrics. Naturally, the performance degrades because of the introduction of the intermediate node between the servers and the receiver. The results show that although OINCV has extra cost when recoding is required, it still outperforms the other methods for all the metrics.



((a)) Coefficients Overhead

((b)) Download Time

((c)) Throughput

((d)) Dropped Packets

Fig. 8: Performance metrics of 0-hops WSN; OINCV vs. $OINCV_{opt}$

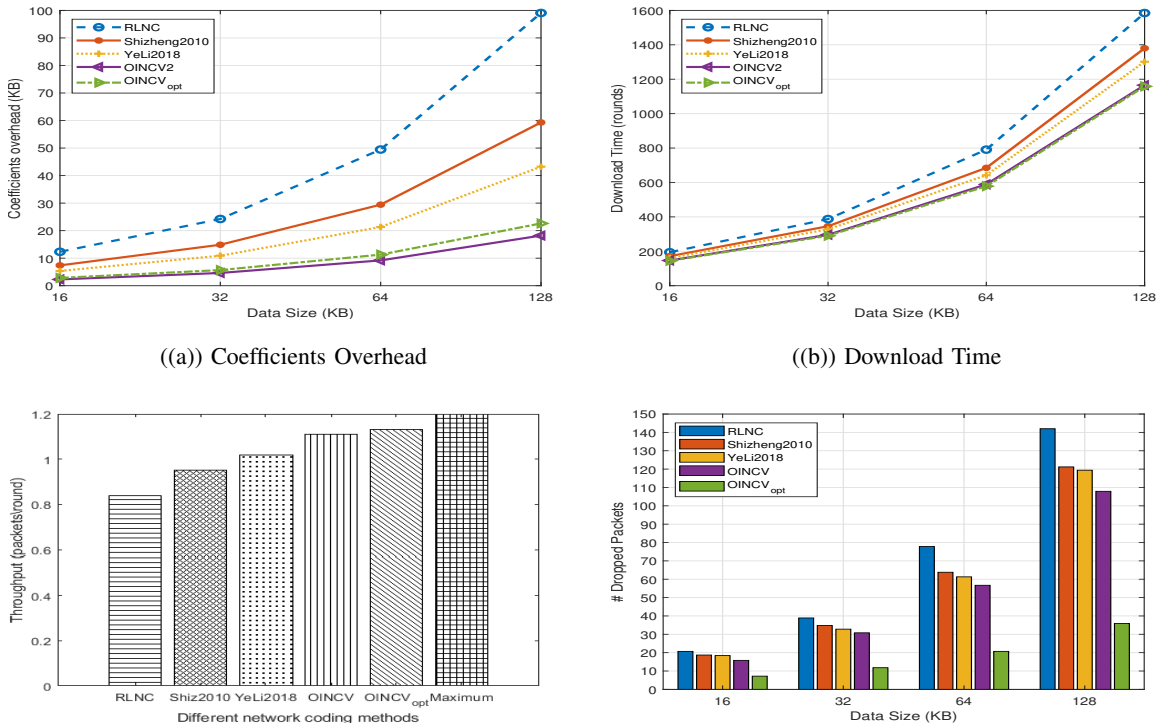
TABLE IV: New Specifications of OINCV and $OINCV_{opt}$

	Packet Payload (Bytes)	Packet Overhead (Bytes)	Number of generations			
			16KB	32KB	64KB	128KB
OINCV2	94	6	6	11	22	44
$OINCV_{opt}$	92	8	2	4	8	15

C. Ethernet Protocol Network

In this case scenario, we consider the following settings:

– File of sizes 50MB, 100MB, 250MB, 500MB are shared by 4 servers to one client who is at zero hops away from the servers. The servers share their packets generation by generation,



((a)) Coefficients Overhead

((b)) Download Time

((c)) Throughput

((d)) Dropped Packets

Fig. 9: Performance metrics of WSN for different network coding vectors methods with recoding.

as in the previous scenario. The client sends an acknowledgement only when it receives all the packets belonging to the respective generation.

– We consider the Ethernet MTU packet size which of 1500 bytes, and the upload rate for each server to be a maximum of 10 packets/round.
– We assume generation sizes of 256 and 512 packets.

Based on these settings and as shown in the previous section, Table V presents the specifications of packet payload and coefficients overhead, and number of generations of each network coding method when the generation size is 256. Figure. 10(a) shows results of the coefficients overhead in MB; the results show that overhead for OINCV does not exceed 1 MB for all the considered file sizes, while the next best method overhead reaches 50 MB. Figure. 10(b) shows the download

time results in terms of rounds. Since the proposed method has the minimum overhead, again fewer packets and generations are utilized for the proposed method and hence less download time. Figure. 10(c) shows throughput results measured in packets/round. The results shows that throughput for OINCV is 39.5 packets/round which is the highest relative to the other methods and the closest to the ideal throughput of 40 packets/round. Figure. 10(d) shows results of the number of dropped packets on the receiver side. Since OINCV has always the minimum number of generations and thus fewer acknowledgements to be sent, it always has the minimum dropped packets.

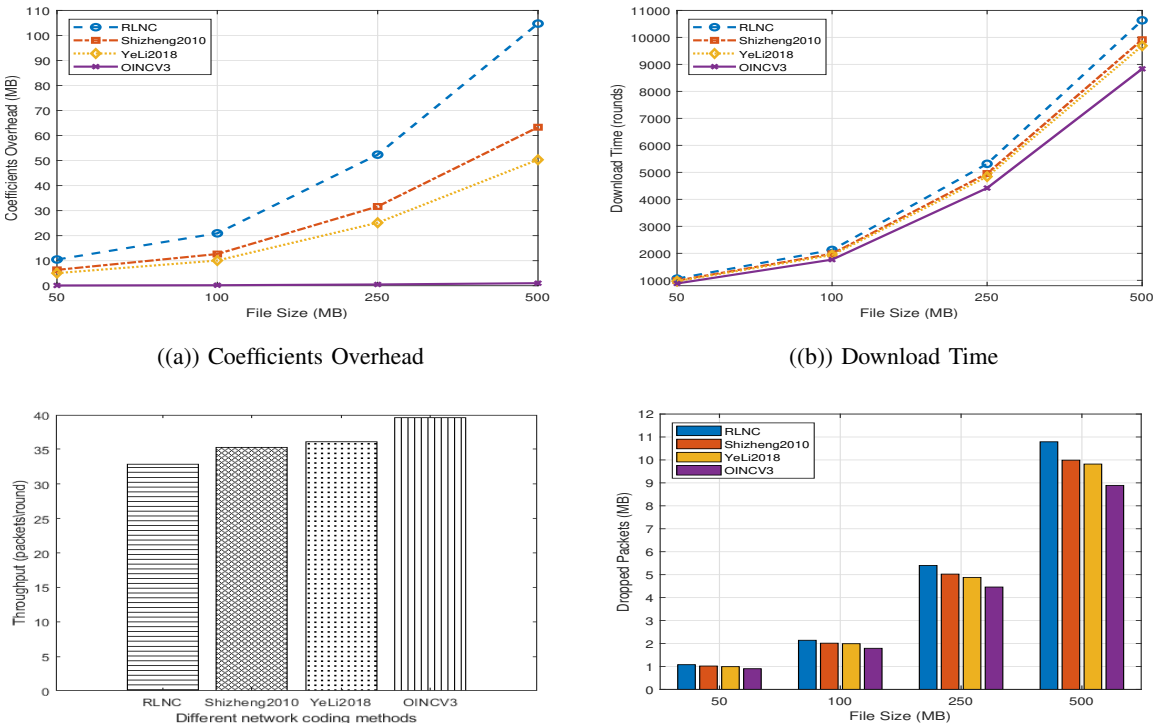
For the generation size of 512, the coefficients overhead for OINCV is maintained fixed at maximum of 1 MB, while it is almost doubled for all the other methods. The download time and throughput are improved for OINCV, on the other hand, these metrics are degraded for the other methods. The number of dropped packets on the receiver side for all the methods is improved because the number of generations is reduced and thus fewer acknowledgements to be shared. These observations are depicted in Fig. 11.

D. Large Scale P2P Content distribution network

In this scenario, we are interested in measuring the overall coefficients overhead and the following settings are considered:

TABLE V: Specifications for different NC methods ($n = 256$)

NC Methods	Packet Payload (Bytes)	Packet Overhead (Bytes)	Number of generations			
			50MB	100MB	250MB	500MB
RLNC	1244	256	165	330	824	1647
Shizheng2010	1334	166	154	308	768	1536
YeLi2018	1365	135	151	301	751	1501
OINCV3	1497	3	137	274	685	1369



((a)) Coefficients Overhead

((b)) Download Time

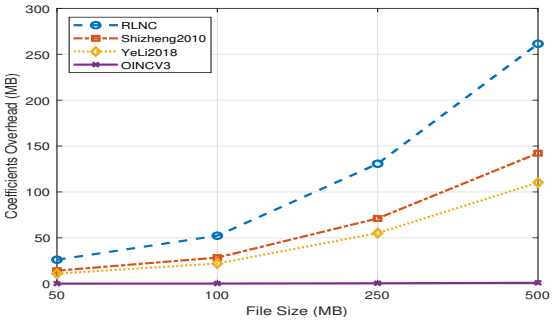
((c)) Throughput

((d)) Dropped Packets

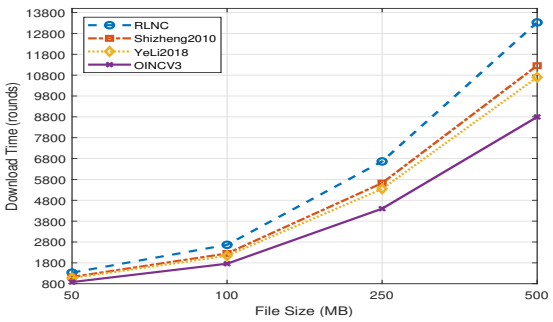
Fig. 10: Performance evaluation of Ethernet network for different NC methods, $n = 256$

- File of sizes 5GB, 10GB, 15GB, and 20GB are shared by 4 servers to one client over a reliable channel. The servers share their packets generation by generation using the sequential scheduling policy. The client sends acknowledgement only when it receives all packets of the respective generation.
- We consider packet size of 16KB.
- We assume a generation size of 1280 packets [5], link bandwidth of 8 MB/round that is shared evenly among the servers such that each server can upload 2 MB/round, while the receiver download rate is set to 8 MB/round.

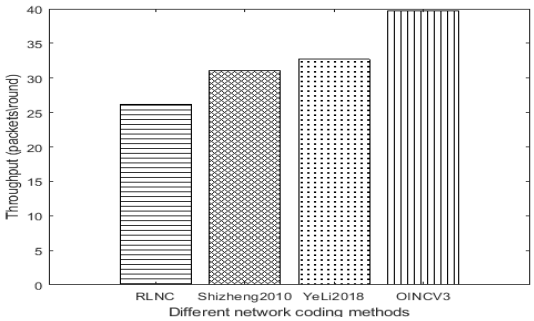
Based on these setting, results of coefficients overhead in MB are depicted in Fig. 12. Since the difference between OINCV and others is huge, log scale is used. The real values are depicted in



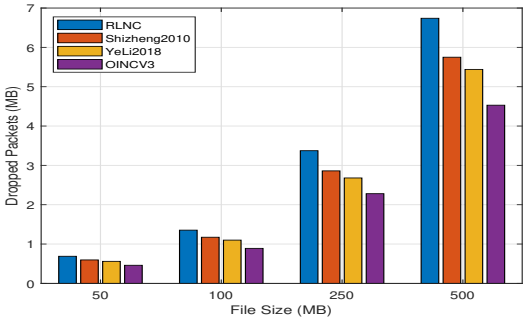
((a)) Coefficients Overhead



((b)) Download Time



((c)) Throughput



((d)) Dropped Packets

Fig. 11: Performance evaluation of Ethernet network for different NC methods, $n = 512$

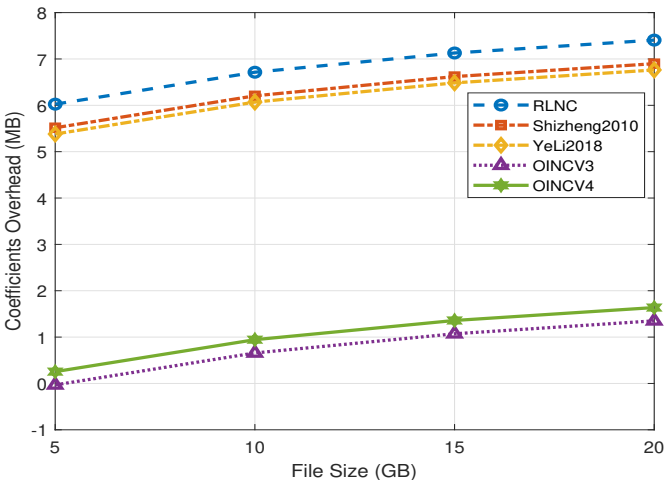


Fig. 12: log scale of coefficients overhead in P2P content distribution network.

Table VI. The results show the big feature of OINCV that is able to reduce the overall coefficients overhead to a little number of megabytes which does not exceed 6 MB even when the file size is 20 GB. Compared to RLNC, [6], and [9], the performance improved by about 99.7%, 99.5%, and 99.4%, respectively.

TABLE VI: Coefficients overhead (MB) in P2P content distribution network.

File Size	5GB	10GB	15GB	20GB
RLNC	413.67	823.8	1247.36	1647.66
Shizheng2010	248.69	495.2	749.88	990.53
YeLi2018	217.18	432.49	654.86	865.02
OINCV3	0.969	1.93	2.92	3.86
OINCV4	1.29	2.57	3.89	5.14

Since the coefficients overhead size per packet compared with payload is small and data link rate is high, other metrics like throughput, and download time are not greatly improved by OINCV for this network case.

V. Conclusion

In this Paper, the problem of network coding coefficients overhead is revisited. A novel approach based on modular arithmetic and prime numbers, and influenced by Chinese remainder theorem (CRT), referred to as One Item Network Coding Vectors (OINCV), is proposed to reduce the coefficients overhead by augmenting only one item coefficient of size 2, 3, or at most 4 bytes to the payload packet. The proposed approach successfully addresses all limitations of the previous methods including the limitations on the generation size and density of the packets within the generation, rocoding on the intermediate nodes, and creating innovative coding vectors. Probabilistic theoretical analysis addresses questions about the proposed method and proves its viability and reliability considering generations sizes that vary from 16 to 6144 packets per

generation. The experimental work is conducted for three different networks types, i.e., Zigbee, Ethernet, and large scale P2P file sharing network. Also, files sizes from 16KB to 20GB are distributed. The results show that the proposed method far outperforms all the previous methods in terms of coding coefficients overhead, download time, throughput, number of dropped packets on the receiver side.

APPENDIX A

PROOF OF LEMMA 1

Proof: By building up the $n \times n$ matrix from the scratch. The first row can be any $n-$tuple vector over $GF(P)$, therefore we have $P^n - 1$ choices , excluding the zero vector. The second row must be linearly independent from the first one and since there are P linearly dependent vectors of the first row, then we have $P^n - P$ choices. For the third row to be linearly independent from the first and the second rows, we have $P^n - P^2$ and so on. In general, for the ith row to be linearly independent from the previous $(i-1)-$rows, we have $P^n - P^{i-1}$ choices. Once we build the matrix by this way, we know that its rows are linearly independent and thus the matrix is invertible. Hence, the number of invertible matrices is given by

$$(P^n - 1) \times (P^n - P) \times ... \times (P^n - P^{n-1}) = \prod_{i=0}^{n-1} (P^n - P^i) \tag{8}$$

The overall $n \times n$ matrices over $GF(P)$ is given by:

$$P^{nXn} \tag{9}$$

divide (8) by (9)

$$= \frac{\prod_{i=0}^{n-1}(P^n - P^i)}{P^{n \times n}}$$

$$= \frac{(P^n - 1) \times (P^n - P) \times ... \times (P^n - P^{n-1})}{P^{n \times n}}$$

$$= \frac{P^n(1 - \frac{1}{P^n}) \times P^n(1 - \frac{P}{P^n}) \times ... \times P^n(1 - \frac{P^{n-1}}{P^n})}{P^{n \times n}}$$

$$= (1 - \frac{1}{P^n}) \times (1 - \frac{P}{P^n}) \times ... \times (1 - \frac{P^{n-1}}{P^n}) = \prod_{i=0}^{n-1} 1 - \frac{P^i}{P^n}.$$

. ■

ACKNOWLEDGMENT

The authors would like to acknowledge the support provided by King Fahd University of Petroleum and Minerals (KFUPM) and the department of Computer Engineering.

REFERENCES

[1] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE transactions on information theory*, vol. 49, no. 2, pp. 371–381, 2003.

[2] L. Keller, E. Atsan, K. Argyraki, and C. Fragouli, "Sensecode: Network coding for reliable sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 2, pp. 1–20, 2013.

[3] M. Jafari, L. Keller, C. Fragouli, and K. Argyraki, "Compressed network coding vectors," in *2009 IEEE International Symposium on Information Theory*. IEEE, 2009, pp. 109–113.

[4] Z. Zhang, "Network coding based on chinese remainder theorem," *arXiv preprint arXiv:1208.3966*, 2012.

[5] A. A. Abudaqa, A. Mahmoud, M. Abu-Amara, and T. R. Sheltami, "Super generation network coding for peer-to-peer content distribution networks," *IEEE Access*, vol. 8, pp. 195 240–195 252, 2020.

[6] S. Li and A. Ramamoorthy, "Improved compression of network coding vectors using erasure decoding and list decoding," *IEEE Communications Letters*, vol. 14, no. 8, pp. 749–751, 2010.

[7] V. Guruswami, "List decoding with side information," in *18th IEEE Annual Conference on Computational Complexity, 2003. Proceedings.* IEEE, 2003, pp. 300–309.

[8] D. Gligoroski, K. Kralevska, and H. Øverby, "Minimal header overhead for random linear network coding," in *2015 IEEE International Conference on Communication Workshop (ICCW)*. IEEE, 2015, pp. 680–685.

[9] Y. Li, S. Zhang, J. Wang, X. Ji, H. Wu, and Z. Bao, "A low-complexity coded transmission scheme over finite-buffer relay links," *IEEE Transactions on Communications*, vol. 66, no. 7, pp. 2873–2887, 2018.

[10] N. Thomos and P. Frossard, "Toward one symbol network coding vectors," *IEEE Communications letters*, vol. 16, no. 11, pp. 1860–1863, 2012.

[11] C.-C. Chao, C.-C. Chou, and H.-Y. Wei, "Pseudo random network coding design for ieee 802.16 m enhanced multicast and broadcast service," in *2010 IEEE 71st Vehicular Technology Conference*. IEEE, 2010, pp. 1–5.

[12] N. Yazdani and D. E. Lucani, "Revolving codes: High performance and low overhead network coding," in *2019 IEEE 2nd Wireless Africa Conference (WAC)*. IEEE, 2019, pp. 1–5.

[13] D. Pei, A. Salomaa, and C. Ding, *Chinese remainder theorem: applications in computing, coding, cryptography*. World Scientific, 1996.

[14] K. H. Rosen, *Elementary number theory*. Pearson Education, 2011.

[15] M. V. Pedersen, J. Heide, and F. H. Fitzek, "Kodo: An open and research oriented network coding library," in *International Conference on Research in Networking*. Springer, 2011, pp. 145–152.

[16] J. L. Alperin and R. B. Bell, "The general linear group," in *Groups and Representations*. Springer, 1995, pp. 39–62.

[17] S. C. Ergen, "Zigbee/ieee 802.15. 4 summary," *UC Berkeley, September*, vol. 10, no. 17, p. 11, 2004.