Performance Improvement of Network Coding for Heterogeneous Data Items with Scheduling Algorithms in Wireless Broadcast

Romana Rahman Ema[1], Md. Alam Hossain[2]*, Nazmul Hossain[3], Syed Md. Galib[4], Md. Shafiuzzaman[5]
Computer Science and Engineering Department, Jashore University of Science and Technology, Jashore-7408, Bangladesh

***Abstract*—This is the age of information. Now-a-days everyone communicates with each other by means of digital systems. Humans are always communicating with each other on the go. On-demand broadcasting is an efficient way to broadcast information according to user requests. In an on-demand broadcasting network, anyone can satisfy multiple clients in one broadcast which helps to fulfill the enormous demand of information by clients. The optimized flow of digital data in a network through the transmission of digital evidence about messages is called network coding. The "digital evidence" is composed of two or more messages. Network coding incorporated with data scheduling algorithms can further improve the performance of on-demand broadcasting networks. Using network coding, anyone can broadcast multiple data items using single broadcast strategy which can satisfy the needs of more clients. In this work, it is described that network coding cannot always maintain its superiority over non-network coding when the system handles different sized data items. However, the causes of performance reduction on network coding have been analyzed and THETA based dynamic threshold value integration strategy has been proposed through which the network coding can overcome its limitation for handling heterogeneous data items. In the proposed strategy, THETA based dynamic threshold will control which data item will be selected from the Client Relationship graph (CR-graph) so that large sized data items cannot be encoded with small sized data items. Simulation result shows some interesting performance comparison.**

***Keywords*—Network coding; scheduling algorithms; CR graph; wireless broadcast; simulation; LTSF; STOBS; performance metric**

I. Introduction

Now-a-days almost everyone carries a portable cellular computing device from a laptop computer to smartphone. All these devices share information to the network on the go. This also requires an infrastructure that does not require a user to maintain a fixed connection in the network and allows mobility. Wireless networks require mobility, distributed sensing and city-wide internet connectivity. For broadcasting the data to the client, network coding uses the limited bandwidth of the wireless efficiently [17] [23]. Network Coding, as a field of study is young which was first introduced in [27] [30]. It is a new concept. Study on the performance of network coding shows that it can utilize the available limited bandwidth of the network to achieve improved throughput in multicast communication [16] [31] [32]. Network coding is applied on on-demand broadcasting network [14] [23] [28]. Here the server broadcasting the data has the information of every client it is broadcasting. Server uses this information to keep track of the data received by clients. Then the server encodes data and broadcasts them on the network. All the clients receive the encoded data and use its own received data to decode the encoded data. Using network coding, a server can serve multiple requests at the same time [17] [22].

Network coding can increase the performance of a broadcasting network in many aspects. It increases throughput, robustness, security in network as well as decreases deadline miss ratio, stretch, response time [17]. But while working on heterogeneous data items, network coding has some drawbacks [9]. It does not perform well as it has been on singular data items [29]. It is caused by the encoding technique which is used in network coding [15]. In XOR encoding, we encode the data items that are found in the maximum clique from the CR-graph [4] [7] [24] [25]. CR-graph is constructed through the data regarding clients' relationships of requested and cached data items [4] [7]. In the proposed THETA based dynamic threshold value integration strategy, the drawbacks of the traditional network coding approach in the scenario of heterogeneous data items have been minimized. Large sized data items and small sized data items have been filtered and encoded separately for improving the performance of network coding.

The rest of this paper is organized as follows. Section II contains related work. Section III illustrates the system model for implementing our proposed strategy. Section IV describes the performance evaluation. Our final thoughts are included in Section V.

II. Related Work

G. G. Md. Nawaz Ali, Yuxuan Meng et al. [1] performed simulation-based analysis based on top of generalized encoding model on both in homogenous as well as the heterogeneous environment for measuring the effectiveness plus adaptability of network coding assisted scheduling algorithms. They analyzed the performance of diverse scheduling algorithms both in non-coding then their proposed coding method utilizing dissimilar performance metrics.

Yuxuan Meng, Edward Chan at al. [2] analyzed the effect of network coding with different scheduling algorithms. They conducted various experiments to measure the performance of broadcasts considering standard access moment, due date ignores relative amount along with typical stretch out.

*Corresponding Author

Cheng Zhan, Victor C. S. Lee et al. [3] proposed a generalized framework so that data scheduling algorithms can be incorporated with network coding for broadcasting on demand requests. They described that with coding, performance can be improved using different scheduling algorithms.

Jun Chen, Victor C. S. Lee et al. [4] proposed a new coding strategy named AC, for implementing an efficient coding mechanism. They also proposed two coding assisted algorithms named ADC-1 and ADC-2 considering data scheduling and network coding. Their simulation results showed that response time was dynamically reduced using both ADC-1 and ADC-2. They also showed that ADC-1 and ADC-2 performed better than conventional and other coding assisted algorithms.

Mohamed A. Sharaf and Panos K. Chrysanthis [8] proposed a new scheduling algorithm named STOBS-α for grouping requests and only one-time delivery of broadcasting results to the clients. Their proposed heuristic on demand algorithm was experimented using access time and fairness for mobile clients.

III. Need of the Improvements

From studies it is noted that when there is no difference in data item size, there is no problem in encoding. For instance, if it is needed to encode three data item d1, d2 and d3 of unit size 1, the size of encoded data item d1⊕d2⊕d3 is also 1. But when we have to encode data items with different size then there is a slight problem. In this condition, the encoded data item's size is the size of the largest item selected for encoding. Let the size of d1 is 1 unit, d2 is 3 unit and d3 is 7 unit. Then the size of encoded data item d1⊕d2⊕d3 is 7 unit. In traditional network coding, large data items are selected with small data items for encoding which in terms cause performance reduction. That also leads to increased stretch and response time, thus hampering the performance of the network [12]. Traditional scheduling algorithms [5] [12] [18] are able to perform better than network coding in such conditions. For this reason, a new modified strategy in network coding has been established to handle heterogeneous data items with ease for maintaining an improved throughput, stretch and response time. The contribution of this paper is as follows:

1) To design a system model, where the server maintains the specification of network coding.

2) To implement the proposed modified strategy which will eliminate the drawback of network coding for heterogeneous data items.

3) To simulate, integrate and analyze our proposed approach with other existing basic scheduling algorithms and compare their performances.

IV. System Model

A. System Architecture

To fulfill more requests earlier than their due dates as well as to assure operative utilization of the constrained bandwidth are the main goals of real-time scheduling and coding. Our system architecture is based on top of the conventional on demand broadcast set-up [4] [7] [10] [14] [18]. The architecture is shown in Fig. 1. The system is set aside by one server with a number of end devices. All end devices have a local cache along with provisions for a certain data core which is broadcasted by the server [1] [13]. Due in the direction of the obligatory room of the end device's caches, a certain guiding principle is applied intended for cache substitution. If the inquired data core cannot be initiated in its cache, the end device sends a request, and its active cache stand-in data to the server through an uplink tunnel [1]. All requests conceivably will necessitate auxiliary data portion from the server. Later than transfer requests headed for the server, end devices listen to the broadcast tunnel to recapture their requested data [1] [13]. It is presupposing that an end device doesn't cache this arriving encoded information but it cannot decode any asked data piece by utilizing this encoded information. If an end device gets and decodes every requested data substance earlier than the time limit, in that case, the requests can be content. In other cases, the request misses its time limit as well as there is no value to the end device [4].

On receiving a request, the server embeds it into a service queue. A request holds up to be scheduled in the service queue until every one of its requested data substances are broadcasted otherwise it gets to be infeasible for scheduling [1] [20]. When the leftover slow-moving phase is smaller than the compulsory phase obligatory towards broadcast every one of the leftover unprocessed data substances, the appeal is considered impossible to be scheduled [1]. A request is removed from the service queue and becomes infeasible, if it misses its required deadline [1]. The server primarily recovers the asked information substance put away within the local database based on top of certain scheduling algorithms then, in that case, encodes the data substance based on data concerning end devices' cached and requested data substance. Lastly, the server broadcasts the encoded information via the downlink tunnel. Inside our model, server and end devices purely exploit the basic XOR operations to encode and decode information [3] [7] [30]. Therefore, the encoding, as well as decoding operating cost and hold-up, can be overlooked.

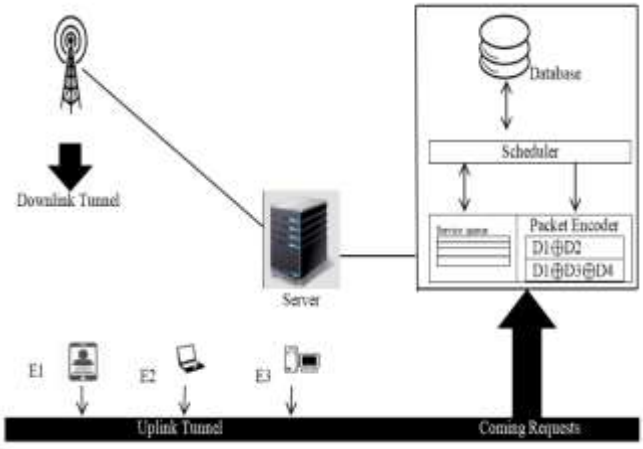


Fig. 1. System Architecture.

B. Graph Model

Our graph model is based on the graph model proposed and discussed by Zhan at al. [3]. In this approach, the CR-graph is constructed on the THETA based threshold mechanism basis. The system has a data server S and n number of end devices E = {e_1, e_2, ... e_n}. Set X(e_i) denotes the set of requested data items of end devices' e_i and set Y(e_i) denotes the set of cached data items of end devices' e_i. The server has a database which contains all the data items. Set m denotes the overall data items contained in the database D. Definition 1: Given E = e_1, e_2, ... e_n, D = d_1, d_2,.... d_m, X(e_i) ⊆ D, Y(e_i) ⊆ D, X(e_i) ∩ Y(e_i) = ∅ [3][6]. A graph G(V, E) can be built the same as follows:

- V= { v_{ij} | end device e_i requests for data item d_j, 1≤ i≤ n, 1≤j≤m}
- E = (v_{i1j1}, v_{i2j2}) | j_1=j_2; or $j_1 \neq j_2$, $d_{j2} \subseteq Y(u_{i1})$, $d_{j1} \subseteq Y(u_{i2})$, | SizeOF($d_1$)- SizeOF($d_2$) | THRESHOLD

If we weigh up on-demand broadcast circumstances in Fig. 2(a) which consists of a server, S and five end devices, e_1, e_2, e_3, e_4 and e_5. Presume that the server has four data substances d_1, d_2, d_3 and d_4. If we assume that end device e_1 has already stored d_2, d_3, d_4 in its cache from preceding broadcasting, end device e_2 has d_1, d_2, d_4 in its cache, end device e_3 has d_2, d_3, d_4 in its cache, end device e_4 has d_1, d_2, d_4 in its cache and end device e_5 has d_1, d_2, d_3 in its cache. Now if we assume that end device e_1 is requesting data item d_1, end device e_2 and e_3 are requesting data item d_2, end device e_4 is requesting data item d_3 and end device e_5 is requesting data item d_4. The data sizes of d_1, d_2, d_3 and d_4 are 1 unit in addition to the broadcast transmission capacity is B=1, which infers the server can broadcast one information piece every time unit.

As of explanation 1, the diagram matching to Fig. 2 is developed as Fig. 3. Within this stature vertex V_{11} speaks to the request from end device e_1 for data d_1. End device e_1 has d_2 requested by e_2 and end device e_2 has d_1 requested by e_1, there is an edge (V_{11}, V_{22}). It is also shown that the end device e_3 has d_3 requested by e_4 and the end device e_4 has d_2 requested by e_3, there is an edge (V_{32}, V_{43}). Other edges are constructed by following the same rule.

Requested Data Item	d_1	d_2	d_2	d_3	d_4
End devices	e_1	e_2	e_3	e_4	e_5
Cached Data Item	d_2,d_3,d_4	d_1,d_2,d_4	d_2,d_3,d_4	d_1,d_2,d_4	d_1,d_2,d_3

Fig. 2. A Demo of On-demand Information Broadcast.

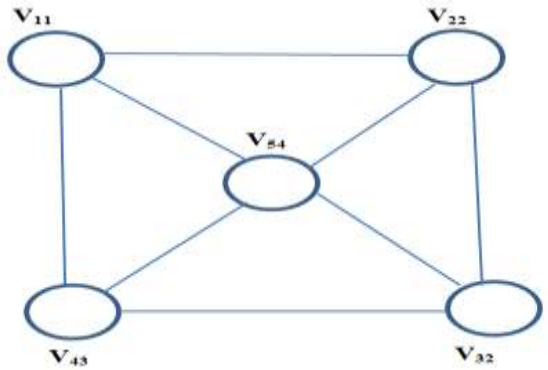


Fig. 3. CR-Graph Developed Commencing the Case in Fig. 2.

C. Proposed THETA based Dynamic Threshold Calculation Strategy

The proposed THETA based dynamic threshold integration is based on the fact that large data sized items will not be encoded with small sized data items.

We construct the undirected graph G (V, E) according to G (V, E) as mentioned in III (B) section. Candidate vertex v_{mi} (denotes data item d_i requested by end device E_m) need to be initiated with V(G).

- For each $v_{ij} \in V(G)$ do
- if SizeOf (d_k) < 100 then
- THRESHOLD= (ln (SizeOf(d_k)))^2
- end if
- if SizeOf (d_k) > 100 then
- THRESHOLD= (SizeOf(d_k) ÷ ln (SizeOf(d_k)))
- End

We need to make a decision which data items would be encoded together. For this reason, we have used dynamic thresholds. This gives better results. Threshold calculating process is given above. Here we have given an example. Suppose we need to broadcast 5-unit data items and 50-unit data items. If we choose 5-unit data items as candidates, the threshold would be around 3. So, 50 unit sized data items won't be in a set with a 5-unit data item set. Only, data item size in between 2 and 8 would come to consideration. Again, when 50-unit data items are candidate, the threshold would be around 15. So, 5 unit sized data item won't be in a set with 50. In this case the range of encoding would be 35 to 65. Though they produce different thresholds, both will face less stretch.

The proposed approach is to change THETA for each candidate by doing THETA= candidate requests requested data item size ÷ 3. But it is tried to choose THETA through a general equation. The proposed algorithm strategy is given below.

- At first, we make pairs which contain Client_Id and Requested_Data_Id.
- Then, we choose each request as a candidate request one by one. For calculating dynamic THETA, we have

used candidate requests requested data item size. Here, THETA is equal to either square of ln (candidate requests requested data item size) or candidate requests requested data item size ÷ ln (candidate requests requested data item size).

- By using THETA, we make a set. If the difference of candidate request's requested data item size and other requests requested data item size is less than THETA then this request is inserted into the set. We repeat this process for all candidate requests.
- From these sets, we find out maximum clique and then do network coding by broadcasting data.

If two vertices of a similar subset are linked through an edge in an undirected graph, it is considered as a clique [4] [26]. There are a number of preferences in the proposed methodology. We are using THETA based dynamic threshold integration. It helps to separate small data items from large data items. If small data items are encoded with large data items, they face large stretch and response time also increases.

V. Performance Evaluation

A. Overview of Comparable Scheduling Algorithm

With the rapid growth of on-demand broadcasting networks, servers have to serve significantly large numbers of clients every day and it is always increasing. So to balance out the increasing load of servers, the necessity of new and improved scheduling algorithms is very high. In network coding, we have to incorporate scheduling algorithms according to our framework so that they maintain their characteristics for scheduling data items for normal networks. Two algorithms have been implemented using the system model (III-A). For heterogeneous data items, STOBS and LTSF scheduling algorithms perform efficiently better than other scheduling algorithms (FCFS, MRF, LMF and others) as those algorithms are generally implemented for single data item [11].

1) STOBS (Summary Tables On-Demand Broadcast Scheduler): In STOBS, the server maintains a queue to store the requests of clients at the time of their arrival. This scheduler chooses a data item for broadcasting with highest (R*W)/S [3] [8] [12] [19]. A summary table T^x is maintained for each request of Q^x. The server keeps the following information [8] [12] [19].

- R: Number of arrival requests for the table $T^{x.}$ When a request for T^x arrives, the value of R increments.
- W: Waiting time of the first request $Q^{x.}$
- S: Table size.

2) LTSF (Longest Total Stretch First): LTSF chooses a data item intended for broadcasting concurring to the order of the maximum total recent stretch [3] [12]. The data piece having the utmost whole current distend is broadcasted earliest [18]. Recent stretch records are calculated by the ratio of the waiting time of pending requests to its time of service [3] [9] [12] [18] [21].

B. Performance Metrics

1) Average response time: Average response time is the ratio of the summation of altogether request's response time in the direction of the entire number of requests [8] [18] [21]. Requests are served quickly if the value of average response time is low.

2) Average stretch: Minimizing the average stretch for heterogeneous data items is the main issue considered for scheduling algorithms. We find average stretch in our simulation model using the following equation.

Average Stretch=Total response time for all end devices/Total service time for all end devices [8] [18] [21].

C. Simulation Model

We performed detailed analysis using CSIM19 [11]. The simulation parameters used for our system architecture (III-A) is shown in Table I. Most of these parameter values are considered from related works [1] [2] [3] [4] [5] [7]. In our simulation model, the server maintains a cache for every end device. At first end devices are generated with data items in their cache automatically. Then, they make their requests to the server maintaining an inter arrival time (IATM). We use IATM in accordance with average data item size. In our simulation:

IATM = 100/Average data item size;

If IATM is low, then end devices will make their requests more frequently which can overload the server.

TABLE I. Simulation Parameters

Parameters	Default	Range	Description
IATM	-	-	Request generation interval
NUMENDDEVICE	100	25-600	Number of end devices
DBSIZE	1000	-	Quantity of information objects in the database
BANDWIDTH	1KB/sec	-	Broadcasting bandwidth
THETA (θ)	0.4	0.2-1.0	Zipf distribution parameter.
CACHSIZE	10	30-180	The maximum amount of data stored in every client's cache.
DWNSIZEMIN	1Kb	-	Minimum size of data item in database
DWNSIZEMAX	60Kb	30-70	Maximum size of data item in database

D. Performance Analysis

The proposed THETA based dynamic threshold integration has been implemented using the system model described in III-A. Overall performance is analyzed and compared using two metrics: average response time and average stretch. We measured the performance by varying item size and cache size with our dynamically changing THETA. Simulation results show that there is significant increase of performance in the network coding environment with our proposed THETA based dynamic threshold

calculation strategy. The reason behind performance improvement is large sized data items are not being selected with small sized data items with our proposed strategy.

1) Performance comparison of average response time for varying item size with different cache size: Tables II, III and Fig. 4(a), 4(b) shows the observations of average response time by varying item size for different cache size. For both STOBS and LTSF algorithms, there is significant increase of performance in the network coding environment. For STOBS, average response time decreases by .25 percent for cache size 30 and .095 percent for cache size 60. For LTSF, average response time decreases by .25 percent for cache size 30 and .26 percent for cache size 60.

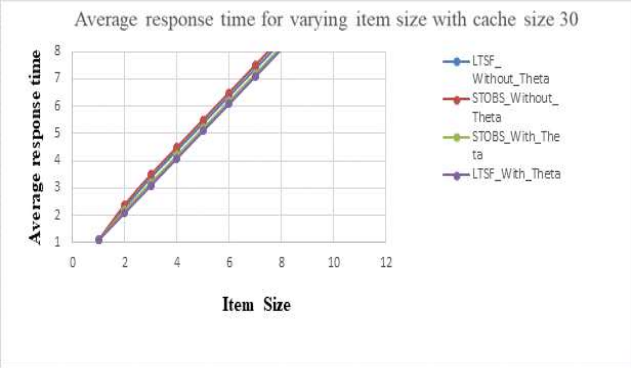
2) Performance comparison of average response time for varying cache size with different item size: Tables IV, V and Fig. 5(a), 5(b) shows the observations for average response time by varying cache size for different item size. For both STOBS and LTSF algorithms, there is significant increase of performance in network coding environments. For STOBS, average response time decreases by .18 percent for item size 60 and .16 percent for item size 30. For LTFS, average response time decreases by .17 percent for item size 30 and .16 percent for item size 60.

TABLE II. AVERAGE RESPONSE TIME WITH CACHE SIZE 30

Serial No.	LTSF Without Theta	STOBS Without Theta	STOBS With Theta	LTSF With Theta
1	1.1	1.1	1.1	1.1
2	2.3	2.4	2.2	2.1
3	3.4	3.5	3.2	3.1
4	4.4	4.5	4.2	4.1
5	5.4	5.5	5.2	5.1
6	6.4	6.5	6.2	6.1
7	7.4	7.5	7.2	7.1
8	8.4	8.5	8.2	8.1

TABLE III. AVERAGE RESPONSE TIME WITH CACHE SIZE 60

Serial No.	LTSF Without Theta	STOBS Without Theta	STOBS With Theta	LTSF With Theta
1	1.1	1.1	1.1	1.1
2	2.3	2.5	2.4	2.1
3	3.4	3.6	3.5	3.1
4	4.4	4.6	4.5	4.1
5	5.4	5.6	5.5	5.1
6	6.4	6.6	6.5	6.1
7	7.4	7.6	7.5	7.1
8	8.4	8.6	8.5	8.1



(a). Average Response Time for Varying Item Size with Cache Size 30.

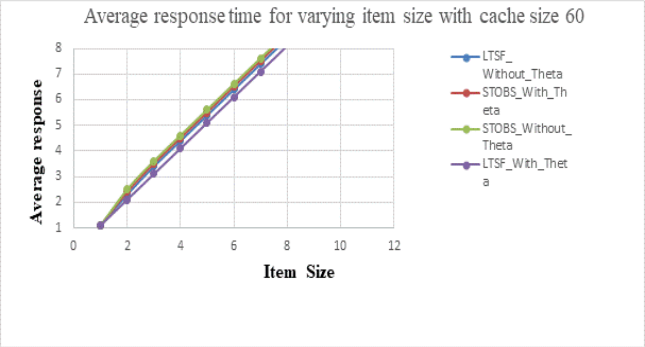


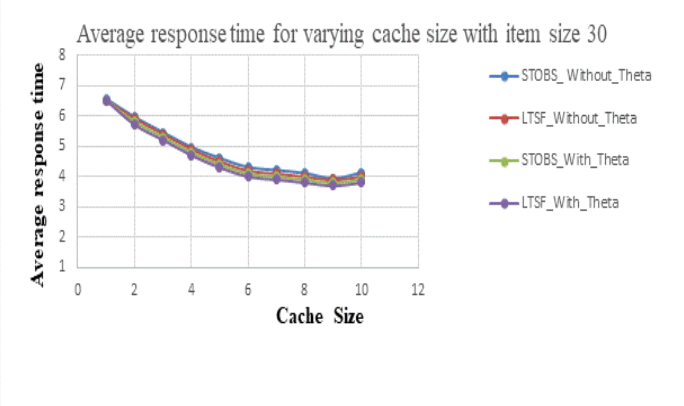
Fig. 4. (b). Average Response Time for Varying Item Size with Cache Size 60.

TABLE IV. AVERAGE RESPONSE TIME WITH ITEM SIZE 30

Serial No.	LTSF Without Theta	STOBS Without Theta	STOBS With Theta	LTSF With Theta
1	6.5	6.55	6.5	6.5
2	5.9	5.95	5.8	5.7
3	5.4	5.45	5.3	5.2
4	4.9	4.96	4.8	4.7
5	4.5	4.6	4.4	4.3
6	4.2	4.3	4.1	4
7	4.1	4.2	4	3.9
8	4	4.1	3.9	3.8

TABLE V. AVERAGE RESPONSE TIME WITH ITEM SIZE 60

Serial No.	LTSF Without Theta	STOBS Without Theta	STOBS With Theta	LTSF With Theta
1	6.5	6.5	6.5	6.5
2	5.9	6	5.8	5.7
3	5.4	5.5	5.3	5.2
4	4.9	5	4.8	4.7
5	4.5	4.6	4.4	4.3
6	4.2	4.3	4.1	4
7	4.1	4.2	4	3.9
8	4	4.1	3.9	3.8



(a). Average Response Time for Varying Cache Size with Item Size 30.

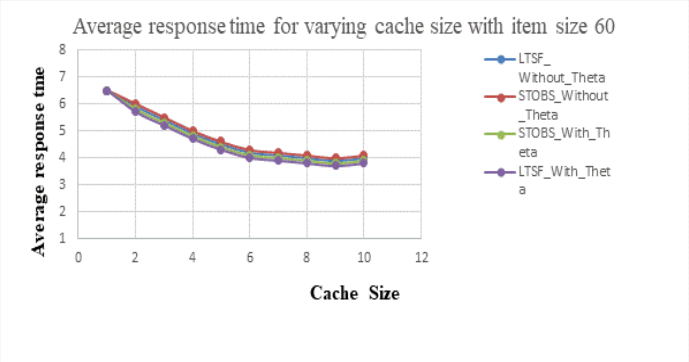


Fig. 5. (b). Average Response Time for Varying Cache Size with Item Size 60.

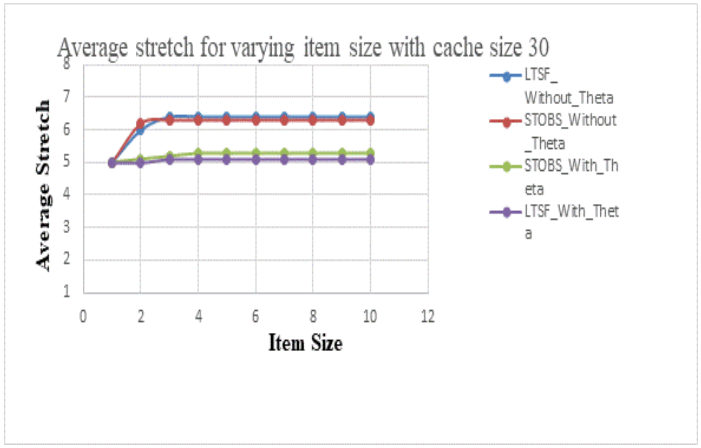
3) Performance comparison of average stretch for varying item size with different cache size: Tables VI, VII and Fig. 6(a), 6(b) shows the observations for average stretch by varying item size for different cache size. For both STOBS and LTSF algorithms, there is significant increase in performance in the network coding environment. For STOBS, average stretch decreases by .91 percent for cache size 30 and .85 percent for cache size 60. For LTFS, average stretch decreases by 1.10 percent for cache size 30 and 1.09 percent for cache size 60.

TABLE VI. AVERAGE STRETCH WITH CACHE SIZE 30

Serial No.	LTSF Without Theta	STOBS Without Theta	STOBS With Theta	LTSF With Theta
1	5	5	5	5
2	6	6.2	5.1	5
3	6.4	6.3	5.2	5.1
4	6.4	6.3	5.3	5.1
5	6.4	6.3	5.3	5.1
6	6.4	6.3	5.3	5.1
7	6.4	6.3	5.3	5.1
8	6.4	6.3	5.3	5.1

TABLE VII. AVERAGE STRETCH WITH CACHE SIZE 60

Serial No.	LTSF Without Theta	STOBS Without Theta	STOBS With Theta	LTSF With Theta
1	5	5	5	5
2	6	6.2	5.1	5
3	6.4	6.3	5.2	5.1
4	6.3	6.1	5.3	5.1
5	6.3	6.1	5.3	5.1
6	6.3	6.1	5.3	5.1
7	6.5	6.4	5.3	5.1
8	6.5	6.4	5.3	5.1



(a). Average Stretch for Varying Item Size with Cache Size 30.

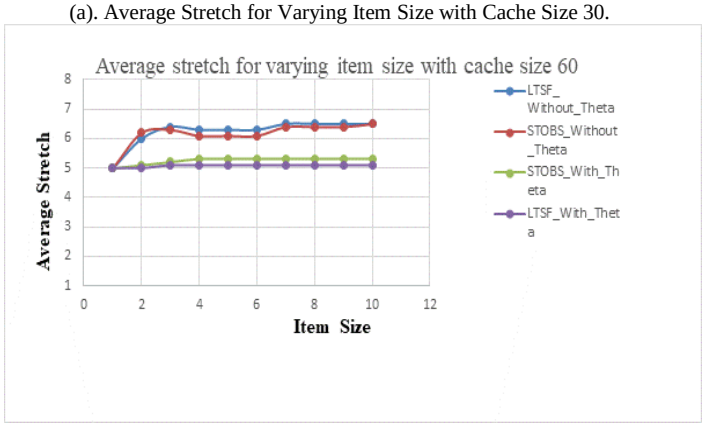


Fig. 6. (b). Average Stretch for Varying Item Size with Cache Size 60.

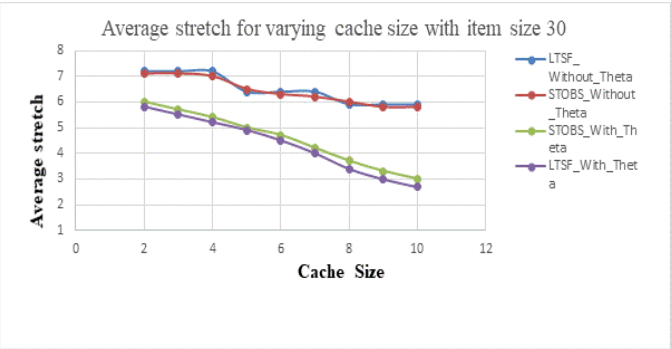
4) Performance comparison of average stretch for varying cache size with different item size: Tables VIII, IX and Fig. 7(a), 7(b) shows the observations for average stretch by varying cache size for different item size. For both STOBS and LTSF algorithms, there is significant increase in performance in the network coding environment. For STOBS, average stretch decreases by 1.75 percent for item size 30 and 2.09 percent for item size 60. For LTFS, average stretch decreases by 2.04 percent for item size 30 and 2.22 percent for item size 60.

TABLE VIII. AVERAGE STRETCH WITH ITEM SIZE 30

Serial No.	LTSF Without Theta	STOBS Without Theta	STOBS With Theta	LTSF With Theta
1	7.2	7.1	6	5.8
2	7.2	7.1	5.7	5.5
3	7.2	7	5.4	5.2
4	6.4	6.5	5	4.9
5	6.4	6.3	4.7	4.5
6	6.4	6.2	4.2	4
7	5.9	6	3.7	3.4
8	5.9	5.8	3.3	3

TABLE IX. AVERAGE STRETCH WITH ITEM SIZE 60

Serial No.	LTSF Without Theta	STOBS Without Theta	STOBS With Theta	LTSF With Theta
1	6.5	7	5	4.6
2	6.4	6.8	4.8	4.4
3	6.3	6.5	4.5	4.2
4	6.2	6.3	4.3	4
5	6.1	6.2	4.1	3.8
6	6	6.1	3.9	3.6
7	5.8	5.9	3.7	3.4
8	5.6	5.7	3.5	3.1



(a). Average Stretch for Varying Cache Size with Item Size 30.

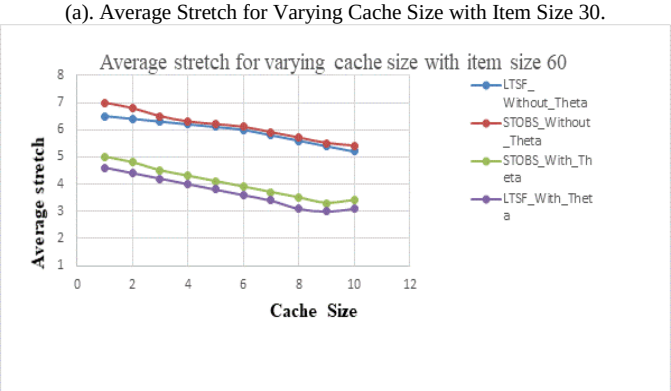


Fig. 7. (b). Average Stretch for Varying Cache Size with Item Size 60.

VI. CONCLUSION

Network coding can be widely used in the on demand wireless broadcast environment. However, it faces encoding problems while handling heterogeneous data items. It fails to provide the best possible solutions when different-sized data substance is encoded jointly. It faces a high stretch. Response time also increases when size differences of different data items are very high. Therefore, in this paper it is attempted to minimize the performance reduction difficulty of network coding in terms of heterogeneous data substance. Based on top of the generalized model proposed and discussed in, a new approach called THETA based dynamic threshold strategy has been introduced for encoding purposes. The proposed approach keeps in mind that large sized data items should not be encoded with small sized data items. The simulation results reveal interesting performance improvement of network coding. STOBS and LTSF scheduling algorithms have been used in this paper and the proposed THETA based dynamic threshold approach has been integrated with these two algorithms. With the proposed strategy, average stretch and average response time is dynamically reduced in a network coding environment. In future, other scheduling algorithms (FCFS, MRF, and LMF) can be integrated with the proposed strategy.

REFERENCES

[1] G. G. Md. Nawaz Ali, Yuxuan Meng, Victor C. S. Lee, Kai Liu and Edward Chan, "Performance Improvement in Applying Network Coding to on-demand scheduling Algorithms for Broadcasts in Wireless Networks", The Ninth International Multi-Conference on Computing in the Global Information Technology, ICCGI 2014.

[2] Yuxuan Meng, Edward Chan and Victor Lee, "Performance Simulation of Network Coding-Based on-demand Broadcast Models", IEEE, 2013.

[3] Cheng Zhan, Victor C. S. Lee, Jianping Wang, and Yinlong Xu, "Coding-Based Data Broadcast Scheduling in on-demand Broadcast", IEEE Transactions On Wireless Communications, Volume 10, No. 11, November 2011.

[4] Jun Chen, Victor C.S. Lee, Kai Liu, G.G.M.N. Ali and Edward Chan "Efficient processing of requests with network coding in on-demand data broadcast environments", ElSEVIER, 2013.

[5] Jun Chen, Kai Liu and Victor C.S.Lee "Analysis of Data Scheduling Algorithms in supporting Real-time Multi-item Requests in On-demand Broadcast Environments", IEEE, 2009.

[6] Cheng Zhan and Fuyuan Xiao "Coding based wireless broadcast scheduling in real time applications", ElSEVIER, 2016.

[7] Jun Chen,Victor C.S.Lee and Cheng Zhan "Efficient Processing of Real-time Multi-item Requests with Network Coding in On-demand Broadcast Environments", 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2009.

[8] Mohamed A. Sharaf and Panos K. Chrysanthis, "On-Demand Broadcast: New Challenges and Scheduling Algorithms".

[9] Md. Ashiqur Rahman, G. G. Md. Nawaz Ali, Yumeng Gao, Syeda K. Samantha, and Peter H.J. Chong "On Accessing Heterogeneous Data Items using Network Coding in Wireless Broadcast", IEEE, 2016.

[10] Jianliang Xu, Xueyan and Wang-Chien Lee "Time-Critical On-Demand Data Broadcast: Algorithms, Analysis, and Performance Evaluation", IEEE Transactions On Parallel and Distributed Systems, Volume. 17, No. 1, January 2006.

[11] H. Schwetman, "CSIM19: A powerful tool for building system models," in Proceedings of the 33th IEEE Winter Simulation Conference, Arlington, VA, USA, 2001.

[12] Xiao Wu and Victor C. S. Lee "Preemptive Maximum Stretch Optimization Scheduling for Wireless On-Demand Data Broadcast", International Database Engineering and Applications Symposium (IDEAS'04), IEEE, 2004.

[13] Jun Chen, Victor C.S. Lee and Kai Liu "On the performance of real-time multi-item request scheduling in data broadcast environments", ElSEVIER, 2010.

[14] Shujuan Wang, Chunting Yan and ZhengtaoY "Efficient Coding-Based Scheduling for Multi-Item Requests in Real-Time On-Demand Data Dissemination", Hindawi Publishing Corporation, volume 2016.

[15] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Me´dard and Jon Crowcrof "XORs in The Air: Practical Wireless Network Coding", SIGCOMM'06, September 11–15, ACM, 2006, Pisa, Italy.

[16] Tracey Ho and Desmond S. Lun "Network Coding: An Introduction".

[17] Christina Fragouli1 and Emina Soljani "Network Coding Fundamentals", Foundations and Trends in Networking, Volume 2, No. 1, 2007.

[18] YiqiongWu, JingZhao,MinShao and GuohongCao " Stretch-Optimal Scheduling for On-Demand Data Broadcasts", 2005 Springer Science + Business Media.

[19] Mohamed A. Sharaf and Panos K. Chrysanthis "On-Demand Data Broadcasting for Mobile Decision Making", Kluwer Academic Publishers, 2004.

[20] Jiun-Long Huang and Ming-Syan Chen "Dependent Data Broadcasting for Unordered Queries in a Multiple Channel Mobile Environment", IEEE Global Telecommunications Conference (GLOBECOM), November 2002.

[21] Miao Wang and Ilias Michalarias "Scheduling On-Demand Broadcast Items".

[22] Demet Aksoy and Mason Sin-Fai Leung "Pull vs Push: A Quantitative Comparison for Data Broadcast", IEEE Communications Society, Globecom 2004.

[23] Marek Konieczny "Network coding in wireless environment".

[24] M. Chaudhry, A. Sprintson, Efficient algorithms for index coding, in: INFOCOM Workshops, 2008.

[25] S. El Rouayheb, M. Chaudhry, A. Sprintson, On the minimum number of transmissions in single-hop wireless coding networks, in: Information Theory, Workshop (ITW'07), 2007, pp. 120–125.

[26] Salim Y. El Rouayheb, Mohammad Asad R. Chaudhry, and Alex Sprintson, "On the Minimum Number of Transmissions in Single-Hop Wireless Coding Networks", 5 July, 2017.

[27] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li and Raymond W. Yeung "Network Information Flow", IEEE Transactions on Wireless Communications, Volume. 10, NO. 11, November 2011.

[28] Xiaojiang Chen, Jingjing Zhao, Dan Xu, Shumin Cao, Haitao Li, Xianjia Meng and Dingyi Fang "Efficient Network Coding with Interference-Awareness and Neighbor States Updating in Wireless Network", Hindawi Wireless Communications and Mobile Computing Volume 2017, Article ID 4974165, 22 pages.

[29] Chen Han, Yuwang Yang and Xu Han "A fast network coding scheme for mobile wireless sensor network", International Journal of Distributed Sensor Networks 2017, Volume 13.

[30] Ali, G. G. Md. N., Lee, V. C. S., Meng, Y., Chong, P. H. J., & Chen, J. "Performance Analysis of On-Demand Scheduling with and without Network Coding in Wireless Broadcast," Future Internet, 11(12), 248, 2019.

[31] Jian Li, Tongtong Li, Jian Ren, & Han-Chieh Chao "Enjoy the Benefit of Network Coding: Combat Pollution Attacks in 5G Multihop Networks," Wireless Communications and Mobile Computing, 2018, pp. 1–13.

[32] K. Lei, S. Zhong, F. Zhu, K. Xu, and H. Zhang, "An ndniot content distribution model with network coding enhanced forwarding strategy for 5g," IEEE Transactions on Industrial Informatics, vol. 14, pp. 2725–2735, 2018.