RAPPORT DE RECHERCHE

L R I

DIGITAL FOUNTAIN CODING WITH XOR OF ENCODED PACKETS FOR BROADCASTING IN WIRELESS MULTI-HOP NETWORKS USING NETWORK CODING

AL AGHA K / KADI N / STOJMENOVIC I

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud – LRI

12/2008

Rapport de Recherche N° 1509

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 490
91405 ORSAY Cedex (France)

Digital Fountain coding with XOR of encoded packets for broadcasting in wireless multi-hop networks using network coding

Khaldoun Al Agha[1], Nour Kadi[1], and Ivan Stojmenovic[2]

[1] LRI, Universite de Paris-Sud XI, Orsay, France {Nour.Kadi,alagha}@lri.fr
[2] SITE, University of Ottawa, Canada ivan@site.uottawa.ca

Abstract. This paper describes CDSOLT, a protocol which optimizes LT code to increase the throughput of flooding in ad hoc wireless network. We propose to apply on each hop and locally XOR operation over encoded packets in decoding process, so that more native packets could be discovered and avoid hence a break in the decoding process. Encoding also can be applied with both received singletons (native packets) and non-singletons, and generated candidates can be used in optimization criteria or in LT coding as random encoded packet from selected number of native packets. The efficiency of network coding is further enhanced by applying source independent backbones. As shown in our simulation, CDSOLT reduces number of transmissions by making use of more computations at nodes. Intermediate nodes may either forward packets without changes over few disjoint routes, or could also apply same algorithm to produce novel encoded packets and assist destination.

Key words: LT Code, Network Coding, Connected Dominating Set

1 Introduction

Digital Fountain codes were proposed recently for channels with erasures in order to avoid feedback channels. The main idea is to send as many as needed encoded packets, in order for each received to decode them and recover the source data. In wireless networks, XOR , abbreviated as $\oplus$, is used as coding operation to control the bit length of transmitted messages. The particular methods differ in the way new encoded packets are generated, and the way they are processed at receiver. Some methods [9], [6] search for an encoded packet that will optimize decoding ability of neighbors by maximizing the number of decoded native packets. We consider here this method, but also method called LT codes, explored also in [5].

We are particularly interested in wireless ad hoc and sensor networks, in applications where multiple data sources broadcast data to all other nodes. That is, in each broadcast, one of the nodes is the source, and the message is to be received, unaltered, by all the other nodes in the network. One particular application of this operation in wireless sensor networks is as follows. [13] proposed partial network coding for data gathering in sensor networks, without data aggregation. The article, however, does not address the preprocessing step, where

every sensor broadcasts its own information to other sensors, so that they can make their encoded packet. This step in itself can be carried by network coding based broadcasting, which is the primary problem being solved here.

Broadcasting using network coding was studied in [4]. However, XOR and opportunistic listening to neighbor receptions was not applied. XOR-based algorithms were explored in [9], [6], [5]. The set of forwarding nodes is restricted to MPR (multi-point relay) set. However, this set is source dependent, therefore particular forwarder node is responsible for some received native packets, but not others, therefore reducing the impact of network coding. Further, all these proposals do not use received encoded packets in future encoding process, and limit the use of already received encoding packets in the process of extracting native packets. Broadcasting with network coding with XOR algorithms was also studied in [8], [3]. The algorithms focus on immediate decoding of each incoming packet and do not refer to any backbone concept. The algorithms also do not use already received encoded packets (only native packets) in future encoding process.

This article makes several contributions to network coding based broadcasting. First, it proposes to apply source independent backbones. Next, we propose to increase the options for coding and encoding, by encoding already encoded packets, and by combining received encoded packets in search for possibly hidden native packet. This will reduce number of needed singletons and speedup decoding process. We propose to apply this idea for broadcasting with network coding in wireless ad hoc networks, to reduce number of transmissions by making use of more computations at nodes.

2 Literature review

2.1 Broadcasting with network coding

Broadcasting using network coding was studied in [4]. Authors apply linear network coding from [2]. Intermediate nodes may combine incoming encoded packets, by making a random linear combination of them. The vector of coefficients (coding vector) over n source packets is transmitted together with the encoded message, and is collected in a decoding matrix G. A received packet is said to be innovative if its coding vector increases the rank of matrix G. Once a node receives n linearly independent combinations, it is able to decode and retrieve the information on n sources. Decoding amounts to solving linear equations with complexity bounded as $O(n^3)$ [2]. The main problem with linear algebra approaches is that the size of linear combination of packets is not controlled. That is, if all packets have L bits, it is not clear how to restrict the size of encoded packets also to L bits. Further, it is not clear how to select and keep small coefficients for linear combinations. These coefficients should be much fewer than L bits, or otherwise the length of transmitted message is much larger than for a single packet. Another problem is that the number of transmitted encoded packets is probabilistic and is not linked to the decoding needs of neighbors. That is, overhearing (opportunistic listening) neighbor receptions is not utilized, as the

algorithm simply assumes that all neighbors need all the packets. It therefore does not address dynamic nature of broadcast communications, and receiving different messages at different stage in the process. These problems, especially associated message size increases, are not discussed in literature. We therefore abandon this approach in favor of those with clear and size controlled encoding operations, such as XOR.

In [10], particular attention is given in deciding probabilities for selecting the number d of packets for XOR-ing. In expectation, the ideal behavior is achieved by the following distribution: $p(1) = 1/K$, $p(d) = 1/(d(d-1))$ for $1 < d \leq K$, where K is the number of native packets to be delivered to neighbors, which are available at given node in either encoded or decoded form. This distribution is used in our experiments. However, this leads to often to situation of having no packet with d=1 in the set, and current decoding process halts in such scenario. For this reason, a complicated mathematical analysis is performed to arrive at probability that is sufficiently higher for $d = 1$, and appropriate for other d values, while maintaining decoding efficiency with small increase in overall number of needed packets. Fountain codes were used in [11] to broadcast n packets from the same source. However in this approach encoding and decoding is done only at endpoints, while intermediate nodes only forward received packets. This problem can be solved by our techniques also, since it is a special case of our more general many to all asynchronous broadcasting process.

Network coding was applied in [9] for deterministic broadcasting. In their XOR-based algorithm, forwarding nodes store only source packets, and also store subset of source packets received by each neighbor (via opportunistic listening). Greedy heuristic is provided to form the encoded packet in which all neighbors can decode received packet. However the effectiveness of network coding is limited by demanding the ability of all neighbors to decode received packet. The algorithm also has little flexibility regarding accuracy of neighbor knowledge and its impact on decoding process. [9] also proposes a Reed-Solomon based coding algorithm. Again, no memorization of received encoded packets is used for combining with future incoming encoded packets. The main drawback of this algorithm is the need to send k packets at once, and the need for neighbors to receive them all to complete decoding, otherwise the whole set of transmitted packets is wasted (even if a single failure happens at a single neighbor). Further, the algorithm cannot incorporate newly received packet into encoded packets after some of k packets were already transmitted. Both algorithms in [9] choose some of nodes to retransmit messages, independent on coding process. We refer to this set as backbone nodes. The proposed backbone construction used in [9] is very similar to MPR (multipoint relay). However the forwarding nodes have to be included in the subsequent packets together with encoded packets, as they differ for each neighbor. In the traditional MPR method, the set of forwarder is fixed for a node, and can be sent by a separate message whose dynamics depends on changes in network topology, but not on neighbors that send messages. This introduces additional overhead to transmitted messages, reducing their reception rate under more realistic MAC and physical layers.

Kadi and Al Agha [6] applied network coding to optimize MPR-based flooding. Each node selects its own MPR (multipoint relay) set which is set of one-hop neighbors that cover two-hop neighbors. Broadcasting from a source node C then proceeds by retransmitting from any receiving node if it identifies that it is MPR node of sender node. Their algorithm has similar goal like XOR-based algorithm [9], to maximize the number of neighbors that can learn one new source packet. However, they do not require the ability of all neighbors to decode received packet in their greedy heuristics. Packet with longest queue delay is added first to the encoded packet. Encoded packets are not memorized at sender or receiving nodes, as they all store only decoded source packets throughout the process. The algorithm from [5] allows nodes to store received encoding packets that couldn't be decoded immediately. Each time a node has a sending opportunity signaled by MAC, it applies LT code over native packets that are in need by at least one of neighbors, and are assigned for retransmission by the node with MPR based backbone. The number of native packets d to be encoded is decided by selected distribution [10]. If $d > 1$ then the receiver will eliminate from the encoded packet all native packets that are present in it and already received. After that, all remaining (and stored) encoded packets will be composed of only native packets which are not individually decoded. If the degree of encoded packet is reduced to 1 then new native packet is recognized, and it can be then used to reduce other, previously received and stored, encoded packets. The receiver then piggybacks IDs of newly recognized packets to the next packet data to inform neighbors.

Algorithms [8], [3] focus on immediate decoding of each incoming packet and assume that sender node is aware of packets already received by it neighbors by some acknowledgements (this is an alternative to overhearing packets used in other schemes). The systematic random network coding algorithm [8] sends every native packets once and afterwards computes random linear combination of all received packets. The opportunistic algorithm [8] native packets are chosen randomly in each iteration for encoding. The latest symbol is added if it remains decodable by all neighbors that were previously able to decode it, otherwise the selection stops. The greedy algorithm [3] first selects a native packet that maximizes the number of neighbors that need it. In the coming iterations, new native packet is added that maximizes number of decoded packets by all neighbors, under the condition that this number does not decrease with respect to the previous iteration. Otherwise the selection stops. This is similar to algorithm [6] described above. The equalizing algorithm [3] chooses, in each step, the neighbor that has the least recovered native packets among those not yet considered. Then the algorithm selects and adds one of native packets that this neighbor has not yet recovered but all the previously selected neighbors can still decode it.

2.2 MPR-CDS: Multi-point relay based connected dominating set

For each message, there exist a dedicated backbone set of nodes that needs to retransmit it so that all other nodes receive the message. This type of backbones

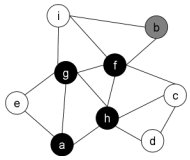

Fig. 1. MPR-CDS algorithm.

is also called connected dominating set. A set V is dominating set for G if each node from G is either in V or has a neighbor in V. We will describe here only one type of connected dominating sets, based on the concept of multipoint relays. It is called here MPR-CDS. Adjih, Jacquet and Viennot [1] proposed a MPR-based algorithm for CDS (MPR-CDS) backbone construction. Each node computes its multipoint relay set by selecting a subset of 1-hop neighbors which cover all 2-hop neighbors. The node attaches the relay list to a 'hello' message which is broadcasted to its neighbors. Upon receiving the 'hello' message, an intermediate node decides to join the CDS if it has either the smallest ID in its neighborhood or if it is the multipoint relay for the neighbor with the smallest ID. Wu [W03] improved the rule by eliminating the node that has the smallest ID among its neighbors, but without two unconnected neighbors. The construction of MPR-CDS backbone requires 2-hop neighbor knowledge, plus a message containing the list of relay nodes of each node. This can be treated overall as CDS construction requiring three rounds of messages, plus another round if the CDS decisions are to be communicated to neighbors. Consider the example in Figure1. Since a and b are the nodes with the smallest ID amongst their neighbors, they decide to belong to the CDS. Node a computes its multipoint relay set $\{g, h\}$ and then attaches the list to its 'hello' message. Upon receiving the 'hello' message, nodes g and h decide to belong to the CDS as well. Similarly, node f decides to belong to the CDS since it is the multipoint relays of b. Finally, nodes $\{a, b, f, g, h\}$ form a CDS by the algorithm [1]. Using the improved algorithm [14], node b is not selected for CDS, and CDS set is $\{a, f, g, h\}$.

3 Contributions

3.1 Source independent backbones

As mentioned, source dependent backbones, like those defined by MPR, have intermediate nodes which retransmit some packets, but not the others. This means that the power of encoding and assisting neighbors is reduced. We propose to apply instead any source independent backbone concept. One such choice is MPR-DS [1] and its improvement [14]. There are other choices, and reader can consult [12] for their descriptions. It is obviously also important that the average number of nodes in the backbone is small, or close to the ones selected by MPR, as otherwise the benefits of shared backbones would be reduced. When applying source independent backbone concept, intermediate nodes are either responsible for retransmitting all or none of packets.

3.2 Faster LT decoding with XOR of non-singletons

We propose to apply XOR operation over S encoded packets in decoding process, so that some new singletons may be generated without any existing singleton in S. For example, XOR of encoded packets $x \oplus y \oplus z$ and $y \oplus z$ will produce singleton x from two non-singletons $((x \oplus y \oplus z) \oplus (y \oplus z) = x)$. This will reduce number of needed singletons and speedup decoding process. We propose to apply this idea for broadcasting with network coding in wireless ad hoc networks, to reduce number of transmissions by making use of more computations at nodes. However, it is clear that proposed enhancement is a general enhancement to LT decoding schemes, for any existing application. In case of wireless ad hoc and sensor networks, the enhancement is especially useful since local computation at nodes and memory space to store results are much cheaper than communication cost, and therefore thousands of such steps could still offset the gain of eliminating even a single transmission. A decoding algorithm applied at receiver node R that does not limit computation and storage cost can be described as follows. Limitations to it can be added when resources are constrained. For convenience, encoded packets can be represented as subsets of set of native packets. For example, $x \oplus y \oplus z$ can be represented as subset {x,y,z}. XOR of two such encoded packets is then corresponding to the symmetric difference of two subsets. For example, if R has two encoded packets $x \oplus y \oplus u \oplus v$ and $x \oplus w \oplus u \oplus t$ then their XOR is $(x \oplus y \oplus u \oplus v) \oplus (x \oplus w \oplus u \oplus t) = y \oplus v \oplus w \oplus t$ which corresponds to symmetric difference $\{x, y, u, v\} \oplus \{x, w, u, t\} = (\{x, y, u, v\} \cup \{x, w, u, t\}) - (\{x, y, u, v\} \cap \{x, w, u, t\}) = \{y, v, u, t\}$. The resulting encoded packet contains native packets that are present in one of packets but not in the other.

3.3 LT encoding from singletons and non-singletons

All existing methods are based on encoding and retransmitting native packets, that is, received and/or decoded singletons or native packets generated at the node itself. However, backbone nodes also store all received encoded packets which they could not decode, but are aware of the elements in the corresponding subsets. They can also be used to generate packet with certain desired degree (number of native packets). For example, if a relay node R has singletons x, u, and subset$\{y, z\}$, then a candidate subsets for encoding are also $\{x, y, z\}$, $\{u, y, z\}$ and $\{x, u, y, z\}$ in addition to $\{x, u\}$. For instance, if LT coding prefers to encode three variables into a packet, in this case it is possible only with the participation of $\{y, z\}$ and joining either x or u. To achieve randomness, appropriate weights need to be calculated. In this case, the probability of selecting $\{y, z\}$ should be 0.5 since they together represent 2 out of 4 possible variables. Further study is needed to identify proper soliton probabilities, that is probabilities for encoding a singleton, or encoding message with d native packets, for each d. These probabilities are expected to be lower for singletons and higher for others, leading to increased network coding benefits and reduction of overall number of transmitted messages.

3.4 NC broadcasting details

The broadcasting process for static networks can follow general framework, as summarized in [12]. Any source starts by sending its packet to all its neighbors. Nodes not in backbone will not retransmit. Nodes in backbone will generate a waiting timeout period whose duration may depend on the number of neighbors in need of a packet. In this more general scenario, waiting period depends on number of missing singletons by neighbors, and even number of received encoded packets. Waiting period can be prolonged/adjusted with reception of further messages. The selection of a formula for timeout duration may impact the performance. Note that similar problem also exists with MPR based approach, and one possible solution is to simply apply random waiting period, which is basically the criterion used in IEEE 802.11 protocol. Neighbor elimination can be applied at backbone nodes if they discover that none of their neighbors is in need of any message. The broadcasting process for mobile networks needs further adjustments. The protocol should have smooth transitions from static to highly mobile networks. One such protocol that can be used as a basis for application in mobile ad hoc networks is described in [7].

4 CDSOLT: Details of the protocol

4.1 CDSOLT overview

CDSOLT is designed to optimize the broadcast traffic in ad-hoc wireless networks or sensor networks using network coding. In CDSOLT, we use a source independent backbone where set of nodes are responsible to forward all the packets which pass through the network. To find such set we use an algorithm which computes the connected dominating set based on multipoint relay [1] and its enhancement [14]. Each node in the network has two buffers, one to keep the received or decoded native packets (d-buffer), another (e-buffer) for keeping certain number of received encoded packets until they could be decoded. Each node in backbone has a timeout which decides how long (number of slots) to wait before it receives slot for the next transmission. The waiting time is computed according to a simple formula (in our simulation, it is proportional to 1/(d-buffer size + e-buffer size). Thus nodes with longer buffers may wait shorter time. Once a slot is received, node then performs the encoding process to combine packets from d-buffer and e-buffer to get an encoded packet of degree d and broadcast it to its neighbors. Degree is the number of native packets which are XORed together to form the encoded packet. The timeout at a node is recalculated each time it receives a packet. On the other hand, the node performs the decoding process when it receives an encoded packet to reduce its degree. However if its degree is still greater than 1 and before inserting it into e-buffer, the node performs the combining process by combining it with other encoded packets stored in e-buffer in order to recover more native packets. Any native packets recovered during these processes are inserted into d-buffer and eliminated from other encoded packets in e-buffer.

4.2 Encoding Process

We use both native packets and encoding packets to form the encoded packet to broadcast. The aim of the encoding process is to find the coding list C that contains d packets, either native or encoded, to be XORed together and broadcast. Any distribution could be used in order to choose d. In our simulation we use ideal soliton distribution described in [10]. In this distribution the p(d=1)=1/k, p(d=i)=1/i(i-1), where $i = 2, 3.., k$ and k is the number of packets available at a node either native stored in d-buffer or included in an encoded packet stored in e-buffer. We choose the d packets sequentially to insert into C. At the first iteration we choose randomly a packet from d-buffer or the encoded packets stored in e-buffer which have degree d. However if the selection was a packet from e-buffer with degree h then it is considered as h native packets, XORed together, to be inserted into C. We continue in the same way until C contains d native packets. Finally the packets in C will be XORed together, their IDs are listed in the header and the result is broadcasted to the neighbors. Consider an example where the buffers at node X are the following: d-buffer=$\{x, y\}$, e-buffer=$\{z \oplus w, w \oplus u \oplus v\}$. Assume $d = 4$, then 4 packets will be chosen to insert it into $C = \{\}$. If i is the number of native packets in C, so the iteration stops when $i = d$. At the 1^{st} iteration let y be chosen then $C = \{y\}$ and $i = 1$; At the 2^{nd} iteration let the encoded packet $w \oplus u \oplus v$ of degree 3 be chosen. Then $C = \{y, w \oplus u \oplus v\}$ and $i = i+3 = 4$. As $i = d$ the iteration stop and the encoded packet will be $y \oplus (w \oplus u \oplus v)$.

4.3 Decoding and Combining process

When a node receives an encoded packet U consisting of d native packet, first it tries to reduce its degree. Using the IDs of the native packets listed at the header, the node tries to retrieve the corresponding packets from its buffer. If any of these packets is found, it is XORed with U which reduce the degree of U by one thus giving U^*. If the degree of U^* becomes 1 then a new native packet P is decoded and inserted into d-buffer. Then P is used to decode other encoded packets stored in e-buffer. During this process, if the degree of any encoded packet becomes 1, we repeat the same procedure for the new decoded packet. If the degree of U^* still greater than 1, then it is combined with other encoded packets at e-buffer. The combining process consist of XORing U^* with each encoded packet V_i stored in e-buffer. If the result of the XOR is a native packet then it is inserted into d-buffer and used to decode other encoded packet. Finally U^* is inserted into e-buffer. To implement these processes we can use one of the following algorithms. For notational convenience, the algorithm will be expressed in terms of subsets. It is applied once between any two receptions of encoded packets; that is, after receiving any new encoded packet.

Decoding and Combining Algorithm

```
Stored in e-buffer: encoded packets (subsets) V1, V2,..., Vp
Stored in d-buffer: decoded native packets;
```

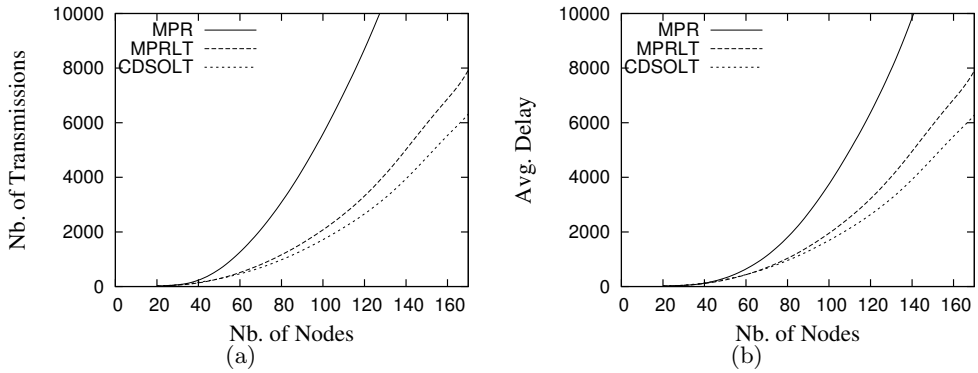

Fig. 2. (a) number of transmissions required to broadcast all the packets into the entire network. (b) average delay for a packet to be delivered to all the nodes

```
1- Apply XOR of each native packet from d-buffer which is part
    of U to eliminate it from U;
2- If U is native packet then place it in d-buffer
3- Else {
        For i=1 to p do {
           Wi=U XOR Vi;
           If Wi is a singleton then native packet is decoded,
                  store it in d-buffer, eliminate it from all
                  other packets in the e-buffer
                  (possibly then revealing more native packets to move
                   similarly to d-buffer)}
        If no native packets is released from 3 then add U to the storage;
        Eliminate duplicates.}
```

Example Suppose that node x has in its e-buffer the following packets $\{v_1 = x \oplus y \oplus z, v_2 = x \oplus u, v_3 = u \oplus y \oplus w\}$. Node x receives the packet $U = u \oplus w$ from its neighbor.

The result of combining process is as follow:
$W1 = U \oplus v1 = x \oplus y \oplus z \oplus u \oplus w \Rightarrow$ "not a singleton" ignore it
$W2 = U \oplus v2 = x \oplus w \Rightarrow$ "not a singleton" ignored it
$W3 = U \oplus v3 = y \Rightarrow$"singleton" store it in d-buffer
The e-buffer of x will be as follow: $\{x \oplus z, x \oplus u, u \oplus w\}$

5 Simulation

In order to evaluate the efficiency of the integration of source independent backbone with optimized LT code for broadcasting in wireless multi-hop networks

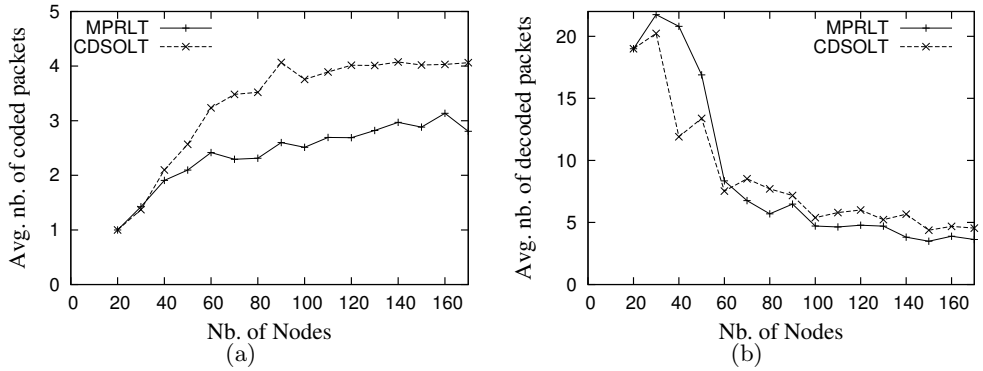

Fig. 3. (a) average number of native packet XORed and sent at each transmission, (b) average number of native packets delivered at each transmission.

we implemented CDSOLT in a custom network simulator written in C++. At the beginning of the simulation, nodes are placed randomly on the simulation area and don't move during the simulation process. Transmissions are received by all nodes within transmission range. A packet transmission takes exactly one time unit. A node can either send or receive only one packet at a time unit. For network traffic, we assume that each node has one packet to broadcast to all the nodes in the network. All of these packets are generated at the beginning of the simulation, and then the simulation continues to run without inserting further packets until all the packets are delivered to the entire network. We assume that only one node can send at a given time unit and in this way we avoid packet collisions. We compare the performance for different number of nodes. We suppose that the average number of neighbors per node is about 20. The nodes are placed randomly in a square network area whose size is chosen according to the number of nodes. The number of nodes changes from 20 to 170. We only consider the connected networks. Our performance metrics are the number of transmissions needed for flooding. To evaluate the memory requirement we calculate the average size of e-buffers and d-buffers. Packets delay is measured by the average time needed for one packet to be delivered to the entire network. We compare the performance of four approaches. The first approach is MPR-based flooding. The second approach, is the original LT code used in flooding where each forwarding node in addition to the source node perform LT code over the packets it has received. The third is MPRLT, our previous approach [KA08b] that combines the Multi Point Relay (MPR) technique with LT code [L02] to perform network coding at each MPR node. The last approach is CDSOLT which combines CDS with optimized LT code, described in section 4.

Figure 2 shows the efficiency of our approach CDSOLT in term of the number of transmissions required for flooding all the packets and the packet delay. Our gain is about 70% in comparison of MPR approach and about 20% regarding to LT code. Even more, we see that CDSOLT gives better performance than our

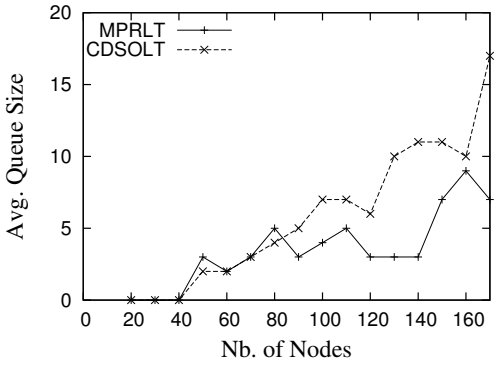

Fig. 4. memory requirement represented by average buffer size.

previous approach MPRLT. This is due to two reasons: First, using source independent backbone (CDS) instead of source dependent backbone(MPR)increases the benefit of network coding by sending more packets at each transmissions as we see from fig3-a because all the packets pass through the same forwarder nodes and this gives more opportunity to perform the encoding. Second, the optimization accomplished in the encoding and decoding function increases the number of packets delivered in single transmission especially for large networks where there are a lot of encoded packets sent and stored in the nodes buffers and thus create more opportunities to release more native packets during the combining process. This is clear from fig3-b. In the other hand, CDSOLT increases the average size of the buffer $avg(|d - buffer| + |e - buffer|)$ for large network as shown in fig4 but this is not a real drawback as nodes have enough memory to handle them because the increment is not too high. In fact the increment in buffer size is due to the optimization done in the encoding function as we choose the combination from singleton and non-singleton. When a node uses packets that it couldn't decode to encode together and send to its neighbors, so its neighbors has less chance to be able to decode and thus they will store more encoded packets in their buffers. This gives the reason of increasing the size of the e-buffer.

6 Conclusion

In this paper we present a protocol CDSOLT to optimize flooding in ad hoc wireless network. In this protocol we integrate two technique: the connected dominating set which is source independent backbone and the technique of network coding represented by one of the fountain code which is LT code. We optimize the coding and decoding process in LT code. We combine the encoding packets locally to release more native packets and thus increase the packets delivered in single transmission. Also LT coding is optimized as we use native packets as well encoded packets to produce new packets to be broadcasted. Our simulation re-

sults show the improvement of the performance when using CDSOLT. For future work we intend to exploit our approach in secure multi path routing.

References

1. C. Adjih, P. Jacquet, and L. Viennot. Computing connected dominated sets with multipoint relays, Oct. 2002.
2. P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Allerton Conference on Communication, Control, and Computing, Monticello, IL*, 2003.
3. R. A. Costa, D. Munaretto, J. Widmer, and J. Barros. Informed network coding for minimum decoding delay. *CoRR*, abs/0809.2152, 2008. informal publication.
4. C. Fragouli, J. Widmer, and J.-Y. L. Boudec. Efficient broadcasting using network coding. *IEEE/ACM Trans. Netw*, 16(2):450–463, 2008.
5. N. Kadi and K. A. Agha. Network coding based flooding using fountain codes. Technical Report 1500, Technical Report 1500 LRI, Univ. Paris-Sud XI, 2008.
6. N. Kadi and K. A. Agha. Optimized MPR-Based Flooding in Wireless Ad Hoc Network using Network Coding. In *IFIP/IEEE Wireless days'08*, Dubai, UAE, November 2008. IEEE Explorer.
7. A. A. Khan, I. Stojmenovic, and N. Zaguia. Parameterless broadcasting in static to highly mobile wireless ad hoc, sensor and actuator networks. In *AINA*, pages 620–627. IEEE Computer Society, 2008.
8. F. D. L. Keller and C. Fragouli. Online broadcasting with network coding. In *4th Workshop on Network Coding, Theory and Applications, NetCod*, Hong Kong, China, Jan. 2008.
9. E. L. Li, R. Ramjee, M. M. Buddhikot, and S. C. Miller. Network coding-based broadcast in mobile ad-hoc networks. In *INFOCOM*, pages 1739–1747. IEEE, 2007.
10. Luby. LT codes. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.
11. R. Kumar and A. Paul and U. Ramachandran. Fountain broadcast for wireless networks. In *IEEE Int. Workshop on Network Sensing Systems*, San Diego, USA, 2005.
12. . D. Simplot-Ryl, I. Stojmenovic, and J. Wu. *Energy efficient backbone construction, broadcasting, and area coverage in sensor networks,*. in: Handbook of Sensor Networks: Algorithms and Architectures (I. Stojmenovic, ed.). Wiley, 2005.
13. D. Wang, Q. Zhang, and J. Liu. Partial network coding: Concept, performance, and application for continuous data collection in sensor networks. *TOSN*, 4(3), 2003.
14. J. Wu. An enhanced approach to determine a small forward node set based on multipoint relay. volume 4, pages 2774 – 2777. Vehicular Technology Conference VTC 2003-Fall IEEE 58th, 2003.