Performance Evaluation of Network Coding: Effects of Topology and Network Traffic for Linear and XOR Coding

Borislava Gajic, Janne Riihijärvi and Petri Mähönen
RWTH Aachen University, Department of Wireless Networks
Kackertstrasse 9, D-52072 Aachen, Germany
Email: {gbo,jar,pma}@mobnets.rwth-aachen.de

***Abstract*— In this work we analyze performance of network coding focusing on two specific network coding schemes: XOR and random linear network coding. We have simulated different topologies and traffic patterns in order to provide better understanding of network coding behavior and its possible bottlenecks. As a part of our performance evaluation we consider also computational complexity of coding and decoding operations which influence packet latency. In particular, we indicate potential drawbacks and trade-offs of network coding when applied on specific topologies under specific circumstances by monitoring the differences in XOR and random linear network coding approaches. Finally, we apply network coding on the topology of existing research network Abilene [1] in order to evaluate network coding performance under realistic conditions.**

I. Introduction

Due to its generality and potential, network coding has received a lot of interest. It has been applied to wide range of communication areas, e.g. information and complexity theory, cryptography, network optimization and wireless communications. Network coding has been envisioned as one possible solution to increase throughput and enable higher data rates than conventional source coding. Additionally, its potential to recover from network failures has motivated research community to investigate different network coding schemes. Early research has mostly focused on theory. Results showed significant capacity gains, but most of the analyzed cases are using relatively simple topologies [2]. Extending these conclusions to larger networks has been done using mathematical abstractions. Due to the broadcast nature of wireless links most of the recent attention has been directed into applying network coding to wireless networks [3]. The seminal theoretical work has been followed by more practical contributions raising numerous questions on implementation constrains [4].

While majority of existing work has been focused on analyzing wireless network coding using algebraic tools, our goal is to evaluate network coding behavior in medium sized networks under more realistic circumstances including point to point connections. In this paper we focus on two commonly considered network coding approaches, namely XOR and linear network coding. We evaluate performance of these approaches under different conditions using ns-3 network simulator [5]. We monitor a number of packets successfully decoded at the receiver side and latency that coding/decoding process introduces, as well as the overall dependency of performance on topology and traffic patterns. This paper is extension of our previous network coding evaluation work [6] containing additional analysis and cases of network coding behavior. We improve our earlier results studying larger networks, new traffic patters, realistic topologies and introducing design constraints at receivers's and sender's sides. The contribution of this paper is in network heuristics especially exploring how different network parameters, including underlying topology, affect the performance of network coding.

The remainder of this paper is organized as follows. In Section II we briefly describe the basic concept of XOR and random linear network coding followed by illustrative examples and related work. Section III explains our main design principles in the implementation of network coding using ns-3 network simulator. We continue by describing simulation results in Section IV and showing how the network coding performance depends on different network parameters. In Section V we give conclusions and outline further research directions.

II. Overview

Following two simple examples provide the basic idea of models we are investigating. These are given as an illustrative examples to provide adequate background for our later discussion.

A. XOR coding

Let us consider the classical scenario where two sides (Alice and Bob) want to exchange a pair of packets via a router, i.e. 4 transmissions are required as shown in Figure 1. First, Alice sends packet to a router, which forwards it to Bob, and then Bob sends packet to the router which forwards it to Alice.

Instead of this, XOR combination of packets is possible at the router side: Both, Alice and Bob send their packets, router XORs them and broadcasts XORed version. After receiving a XORed packet, A$\oplus$B, both Bob and Alice are able to decode the packet sent from the other side by simple XORing the received packet with their own. This is illustrated in

Manuscript received April 30, 2009; revised June 14, 2009; accepted September 3, 2009.

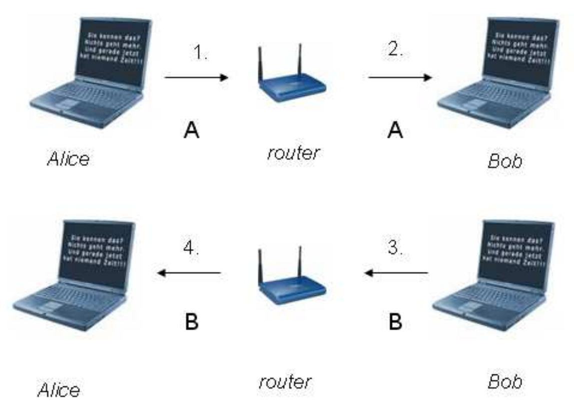

Fig. 1. Exchange of packets without network coding.

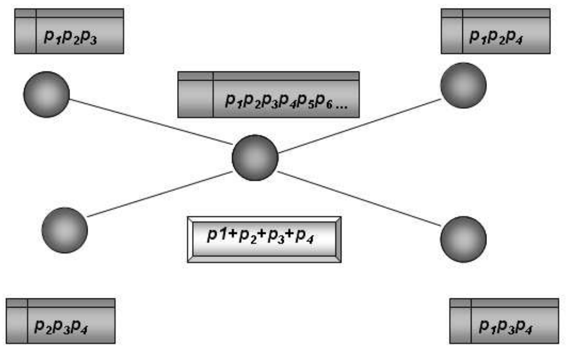

Fig. 3. Maximizing coding gain by making combinations of n packets if all receivers already have $n-1$ packets of the same combination.

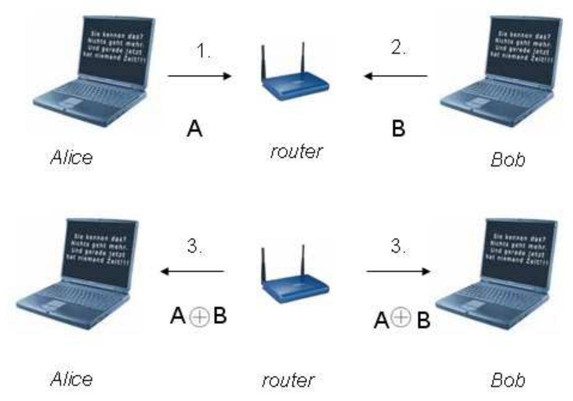

Fig. 2. Exchange of packets using network coding.

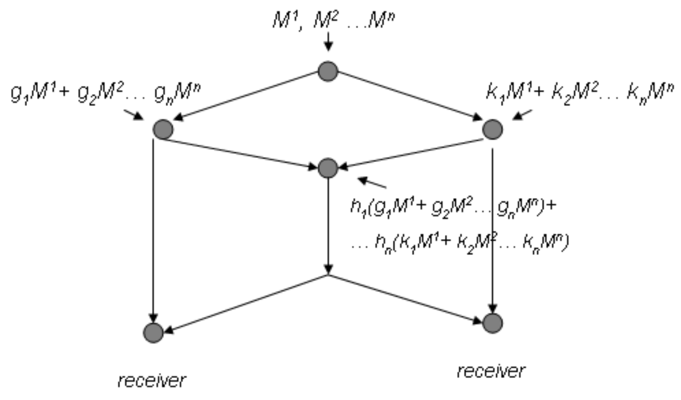
$M^1, M^2 \ldots M^n$
$g_1M^1+ g_2M^2\ldots g_nM^n$
$k_1M^1+ k_2M^2\ldots k_nM^n$
$h_1(g_1M^1+ g_2M^2\ldots g_nM^n)+$
$\ldots h_n(k_1M^1+ k_2M^2\ldots k_nM^n)$
receiver
receiver

Fig. 4. An example of linear network coding, where $M^1, M^2...M^n$ are source packets multicast to the receivers, and coefficients g_i, k_i and h_i are randomly chosen elements of a finite field.

Figure 2. Moreover, encryption is achieved by the fact that it is impossible to reverse the operation (decode message) without knowing value of one of two initial arguments.

In more complex scenarios, such as shown in Figure 3, a router will have different coding possibilities, among which it should choose the one that maximizes the number of packets delivered in one transmission. In this particular example, router having packets $p_1p_2p_3p_4p_5p_6$... will optimize decoding at the sides of receivers. Thus all of them will be able to decode missing packet, combining $p_1p_2p_3p_4$ together. This leads to simple conclusion: Router will maximize coding gain by making n packets combinations if all recipients already have $n-1$ packets of the same combination. In order to optimize XOR network coding gain in our implementation, we will be essentially following this rule, while taking other objectives into consideration.

B. Random linear network coding

Random linear network coding approach is, in general, similar to XOR coding with the difference that the XOR operation is replaced with linear combination of data (in essence, matrix multiplication) where coefficients of linear combination are taken from a certain finite field. This provides more flexibility in how the packets can be combined. Successful reception of information does not depend on receiving a particular data packet but on receiving sufficient number of independent packets.

Let $M^1, M^2...M^n$ denote the original packets generated by several sources, also called native packets [7]. The encoded packet would be a linear combination of $M^1, M^2...M^n$ with associated set of coefficients $g_1, g_2, ...g_n$ from a certain finite field F which implies that it has a form of $X = \sum_{i=1}^{n} g_i M^i$. In other words two vectors exist: first, $g_1, g_2, ...g_n$ -encoded vector, which is used at the receiver side to decode the message, and second, $X = \sum_{i=1}^{n} g_i M^i$ - information vector [8]. Encoding can be performed recursively, with already encoded packets, as shown in Figure 4.

In order to retrieve original message decoder has to solve the system of m equations $X^j = \sum_{i=1}^{n} g_i^j M^i$ using Gaussian elimination algorithm, where unknowns are M^i. This system with m equations has n unknowns, and having $m \geq n$ is prerequisite for decoding. Fulfilling this requirement is not a guarantee that the message will be decoded since some of the linear combinations might be linearly dependent.

C. Related work

Notion of network coding, where intermediate nodes are allowed to encode and decode messages was introduced by Ahlswede et al. [2]. This work focuses on discussion of the maximum throughput possible in a directed network for a multicast session between a source and a given set of receivers, and presents benefits in increasing the throughput by using network coding. Koetter and Médard [9] were considering network capacity problem entirely on analitical framework using linear codes. Recent work on improving throughput of wireless network using XOR coding [3], [4] has introduced more practical approach and showed high benefits of XOR packets combination layer between IP and MAC. This work follows more reliable approach using backoff and ack mechanism of modified 802.11 unicast. In order to avoid dependency between nodes in coding/decoding process, random linear network coding has been proposed as one of the most promising solutions. Nodes execute independently random linear transformations of input data over certain finite field [10]–[12]. Applying network coding to probe packets improves network tomography in terms of link loss rate estimation and bandwidth efficiency [13]. Introducing limits in decoding delays for specific applications, e.g, streaming powered with packet decoded acknowledgements from receiver's side leads to higher optimization of the throughput and better performance of network coding [14]. Sundararajan et al. [15] proposed the idea of network coding layer between the network and transport layers. This approach followed by adaptation of current TCP acknowledgement mechanism to better serve network coding requirements shows improvement in network coding results without large changes in current network structure.

III. Implementation and Scenarios

In our implementation, network coding/decoding process is inserted on the top of existing network layer, due to availability of information about packet source and destination necessary for XOR coding/decoding that IP header contains. We are exploiting helper functionalities of ns-3 to build point to point links between nodes in the network, with a desired data rate and delay. Nodes have buffers of infinite size for storing packets. Every node keeps track of received and sent packets, based on which it gains knowledge about packets distribution in the neighborhood. Thus, during coding/decoding every node relies on local information about overall packet distribution, without contacting other nodes. This information is of high importance for optimization, since every packet combination will not lead to successful data recovery and network coding benefit. We aim at maximizing this benefit following the rule illustrated in Figure 3 only if this opportunity exists. Therefore, each packet has to be created in such a way that every neighbor after receiving packet combination and performing XOR operation with packet it posses, retrieves a missing packet. This is possible only if node, receiving N packets combination, already owns $N-1$ packets of received combination. In our particular case, node having M point to point interfaces (neighbors), in order to benefit from broadcast will make M packets combinations, ensuring that every neighbor already has $M-1$ native packets of that combination. Before combining, the node performs look up in its received packets buffer finding all packets with $M-1$ sources (packets with same unique ID but with different source address). Such packets, having the same ID but $M-1$ different source addresses denote the same packet received from $M-1$ different sources, i.e., it is one possible packet to be put into final combination. At the end, the node makes all possible M (number of neighbors) combinations out of packets that have $M-1$ sources, and broadcasts it. Information about combined packets is stored in metadata of packet combination to facilitate decoding process. Upon receiving the packet, the node performs look up in the buffer one more time to find combination of packets that has to be XORed with received combination in order to retrieve missing packet. Validity of resulting packet is checked performing checksum over payload [16].

Our implementation of XOR coding/decoding at the sender/receiver side is seen as auxiliary process to "common sending/receiving". Thus after, e.g., receiving the packet, node first checks if incoming packet is combination, if not it performs ordinary processing of packet like in the case that no coding/decoding mechanisms are present. In the case of sending the packet, the node first checks if there is possibility to send a packet combination based on information it has about the packet distribution in its neighbourhod. Otherwise, it sends packets following regular pattern. In other words it is opportunistic, because it performs coding operation when such possibility arises.

On the other hand, random linear network coding is simpler due to the fact that node does not need to have overall knowledge of its surroundings, since packet generation is carried out probabilistically. Our implementation of this model relies on ns-3 random number generators for creating coefficients and selecting packets to be combined. A node generates output packet as a linear combination of randomly chosen packets from buffer and random coefficients $g_1, g_2, ...g_n$. Structure of packet combination is recorded in the form of vector of coefficients and packet IDs and stored in the metadata. Based on this information decoding side is aware of each packet composition and performs Gaussian elimination after sufficient number of combination packets of same structure is received (n packets with linearly independent vector of coefficients). Overhead that this additional information introduces is negligible compared to the packet size.

Our network coding scenario resembles constant bit rate datagram communication between nodes. In our experiments capacity of point to point links is initially set to 4.5Mbps, with the delay of 5ms. Network topology varies based on

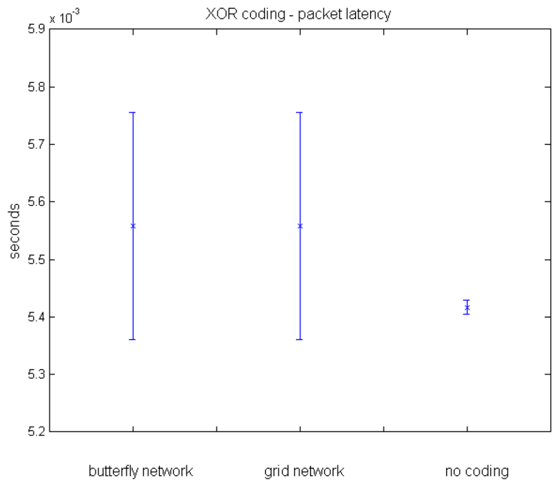

Fig. 5. Mean latency and standard deviation for XOR coding (butterfly and grid networks) and no coding.

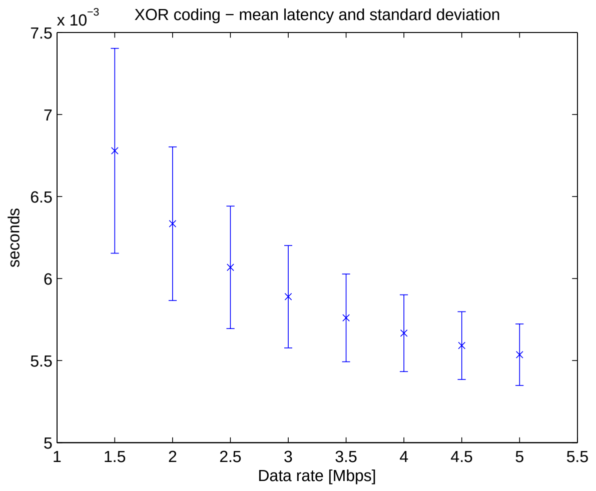

Fig. 6. Mean latency and standard deviation for XOR coding over different data rates.

the number of nodes involved ranging from 25 to 121 nodes and the type of connectivity (grid or butterfly networks). We additionally exploit the case of randomness in building network topology by applying certain probability of link existence between two nodes. Finally, we observe network coding performance applied to realistic networks by studying the performance in the Abilene topology. Basic traffic model relies on ns-3's OnOff Application, which switches transmission and idle states according to the predefined On and Off intervals. During the On interval, CBR traffic with defined data rate of 448Kbps and packet size of 210 bytes is generated, while Off interval denotes no traffic. Similar to the randomness applied in building different topologies we create random traffic patterns by varying transmission probability of nodes in the network.

IV. Results

Large part of obtained results are based on measuring network coding impact in terms of packets that are additionally decoded at the receiver side compared with the case without coding. We are varying data rate, packet delay, traffic patterns as well as the underlying topology and monitor impact of these parameters on overall results. Different traffic patterns are obtained by altering transmission probability for every node in the network. For underlying topology we choose butterfly and grid networks with the number of nodes ranging from 25 to 121, extending our observations with the random topology case generated by varying connection probability between nodes. Results show that benefit of both types of network coding highly dependents on various parameters.

One of important concerns of network coding implementation is the mean packet latency that coding and decoding introduce. Increasing the size of packet buffers enlarges the amount of data that each node has to be aware and to process leading to higher decoding delays. For XOR coding, having a relatively small number of packets P with $M-1$ sources in

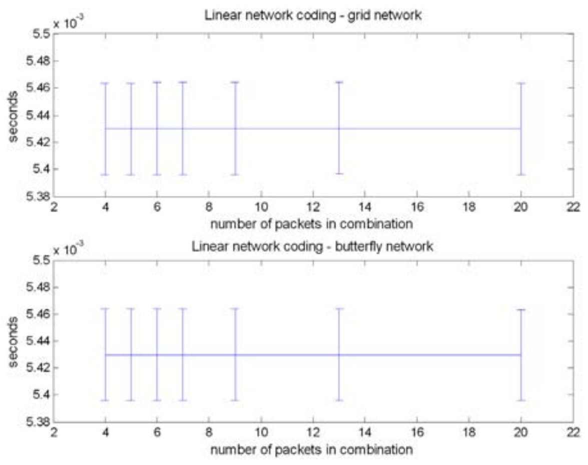

Fig. 7. Linear network coding - mean latency and standard deviation (grid and butterfly networks).

the buffer, combination of N (number of interfaces) over P becomes very large (due to $P!/N!(N-P)!$), and the task of finding combination which results in benefits for all neighbors, time consuming process. On the other hand, if the size of packet buffer is not sufficiently large the node will not have enough information based on which it can perform beneficial coding. Our results show that for network settings we observe latency caused by network coding is negligible for most of the applications. The difference in packet latency when XOR coding is applied over 36 nodes butterfly and grid networks compared with the case of no coding is illustrated in Figure 5.

Results show that latency is not heavily dependent on the underlying topology, but primarily varies based on the channel parameters; data rate, and delay it introduces. For the fixed channel delay, latency added by XOR coding decreases with increasing the data rate of transmission as shown in Figure 6.

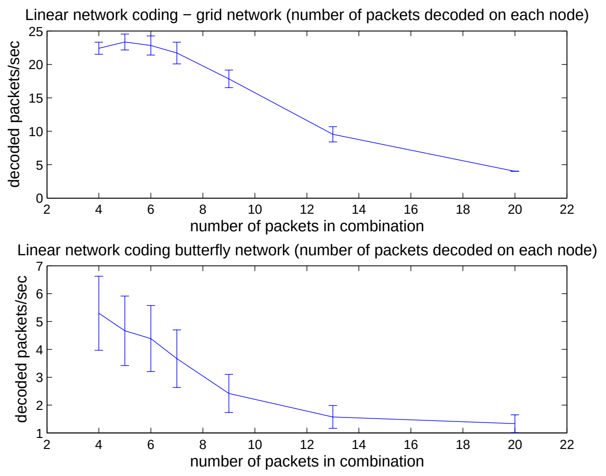

Fig. 8. Linear network coding - mean number and standard deviation of packets decoded on each node (grid and butterfly networks).

Evaluation of linear network coding latency and comparison with XOR coding case is done by repeating the tests under the same network settings as in the XOR case and fixing the data rate to 4.5Mbps. Additionally, we monitor the impact of packet combination size on coding/decoding delay. Our results show that in the terms of latency linear network coding performs better compared to XOR coding for the same channel parameters.

Lower delay occurred applying linear coding for data rate 4.5Mbps and fixed channel delay as shown in Figure 7. Moreover, Figure 7 demonstrates influence of packet combination size on latency. For relatively small packet combinations, e.g. containing less than 20 packets, Gaussian elimination does not introduce additional computational delay due to larger number of native packets combined. On the other hand number of packets combined and underlying topology have impact on linear network coding performance in terms of decoding gain. For particular topology, variation of packet number in combination leads to different decoding outcome as illustrated in Figure 8. Increasing the number of packets put into the combination might lead to performance degradation due to higher complexity and longer time needed for sufficient number of coded packets to be received. In both cases, XOR and linear network coding results rely on packets distribution and the number of packets each node stores in its buffer. In our implementation we assume that nodes have unlimited capacity buffers for collecting packets. Thus, every received packet is stored in the buffer and potentially influences decoding outcome. Based on packet buffer size at the time of coding/decoding decision network coding gain varies. In order to better address packet diversity and buffer assets influence on decoding outcome keeping the buffer sizes unlimited we compare two network cases sharing the same network parameters set but differentiating in packet resources of nodes participating in communication.

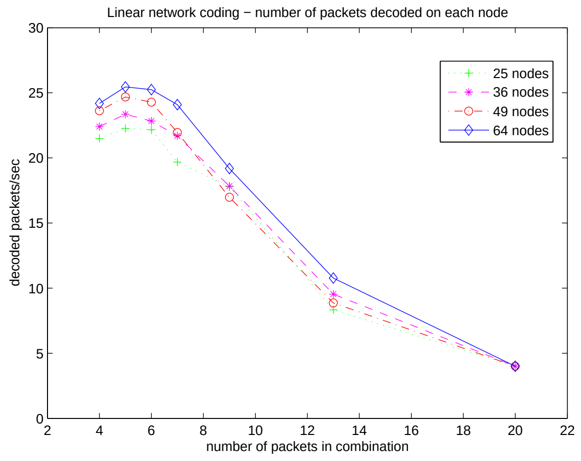

Fig. 9. Linear network coding - influence of network size and buffer size on number of packets decoded

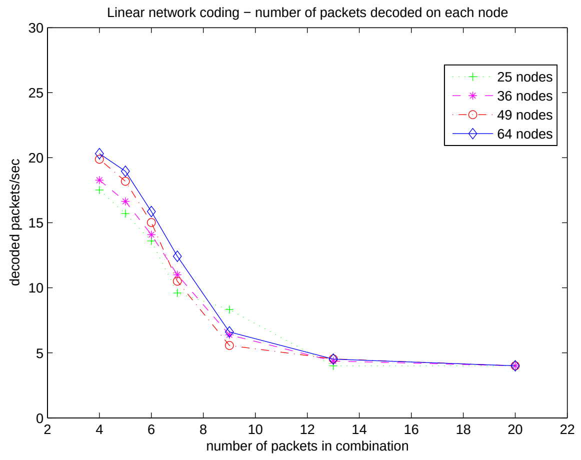

Fig. 10. Linear network coding - influence of network size and buffer size on number of packets decoded (30% less packets in buffers compared to the network setting from Figure 9)

Figure 9 and Figure 10 show the difference in decoding gain for grid networks containing 25, 36, 49 and 64 nodes while node buffers of network from Figure 10 contain 30% less packets than the network in Figure 9. Nodes having 30% more packets stored in buffers are able to decode much less than 30% additional packets, thus the number of decoded packets on each node is not linear function of buffer size. Nevertheless, having larger packet buffers gives more combining opportunities to nodes and ensures higher network coding benefit. Moreover, the shape of decoding curve, thus the decision upon optimal packet combination size for the given topology settings, varies based on the node's buffer size, as well.

Another interesting objective to investigate is relation between buffer limitations and decoding gain. Initial step in our evaluation work is to limit buffers to particular size. In the case of receiving the packet after the buffer has reached its

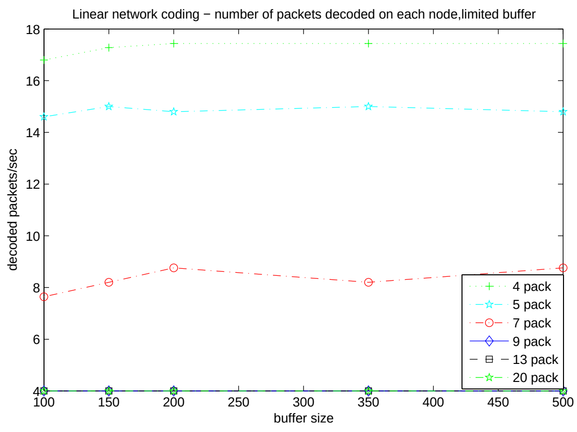

Fig. 11. Linear network coding - influence of the buffer size and combination size on the number of decoded packets.

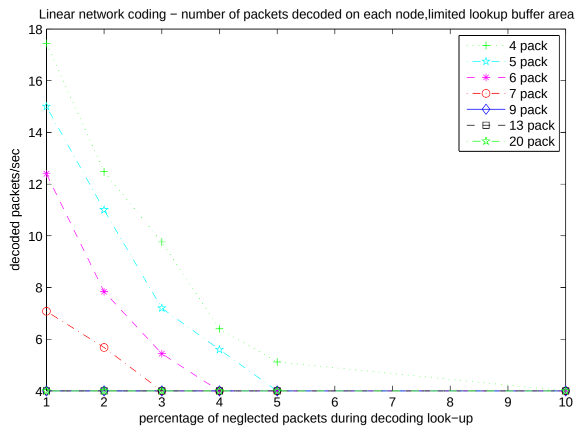

Fig. 12. Linear network coding - influence of limiting decoding look-up area in buffers on the number of decoded packets.

limit, the first enqueued packet in the buffer is deleted in order to make the space for incoming one. First part of our buffer limitation evaluation is in monitoring influence of buffer limit to decoding gain. Thus, we examine five different buffer sizes with 100, 150, 200, 350 and 500 packets. Results show slight increase in the number of decoded packets due to the increase of buffer size. Applied to the same packet combination size larger packet buffer does not bring significant decoding gain. Smaller packet combinations perform better like in the case of infinite buffers as illustrated in Figure 11. This raises the question of larger buffer's utilisation and justification of applying them in network coding scenarios. While introducing additional processing costs, infinite buffers do not provide fundamental enhancement of network coding performance.

On the other hand there is large difference in decoding gain while limiting lookup area at the receiver's side. Not taking into account even less than 5% of available packets in the buffer during decoding process heavily changes decoding outcome. Neglecting more than 10% of packets during the decoding lookup phase deteriorates network coding performance dramatically. Hence, the decoding gain does not justify implementation of network coding as illustrated in Figure 12. Based on this observation we can consider existence of relatively small amount of packets heavily influencing decoding gain. This remark might have larger impact in the case of introducing mechanisms for faster packet look-up with the possibility of false negatives. Disregarding even few packets during decoding look-up due to false negatives will cause poor decoding results. The size of the network is another important parameter influencing decoding results. Increasing the number of nodes involved in communication is followed by increasing the number of packets decoded, for the same traffic and topology parameters applied. Figure 13 illustrates the impact of network size on number of packets decoded on each node for grid networks built upon 36, 64, 81, 100 and 121 nodes.

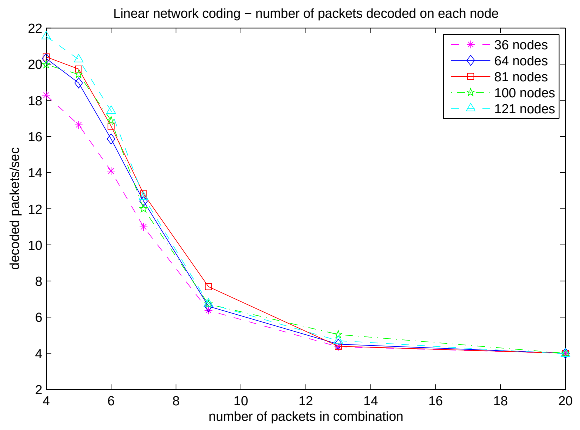

Fig. 13. Linear network coding - influence of network size on the number of decoded packets.

A. *Random traffic and topology settings*

Changing the traffic pattern by not necessarily involving every node in the communication, influences overall performance of network coding in terms of efficiency while packet latency remains the same. Defining the certain probability for each node to broadcast data determinates its participation in packet transmission as well as in network coding. In our implementation we examine relation between given broadcast probability and network coding performance. Figure 14 shows results for 90% broadcast probability where mean number of decoded packets is proportional to this predefined probability (approximately 90% of network coding benefit compared to the case where all nodes broadcast).

Standard deviation of decoded packets number is high due to the large possible variations of number of sent and received packets on the same node as illustrated in Figure 15. It shows the mean difference between the number of packets received and sent on each node while high positive and negative peeks of standard deviation represent two extreme cases that might

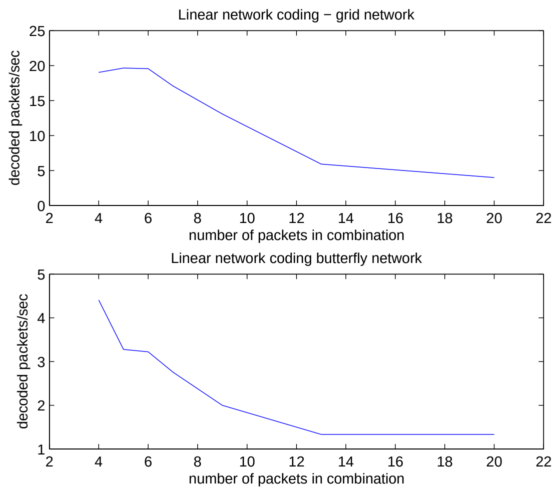

Fig. 14. Random traffic pattern (node broadcasts with 90% probability) mean number of decoded packets

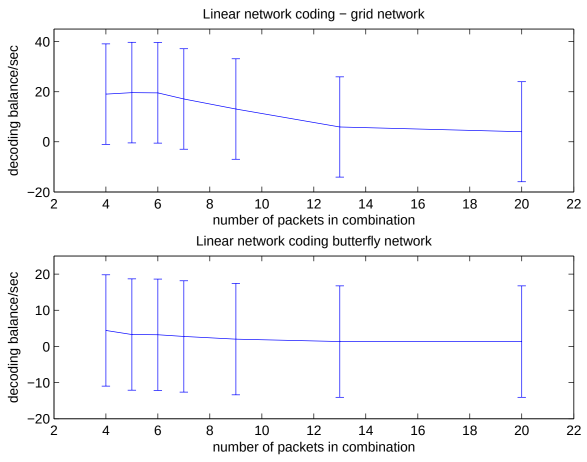

Fig. 15. Random traffic pattern, mean number and standard deviation of decoded packets (node broadcasts with 90% probability)

happen in random traffic scenarios. High positive values are reached when node is not sending packets, since it is always able to receive and decode, thus the difference between number of received and sent packets is exactly the number of received and decoded packets. On the other hand negative extreme is obtained when the node is sending packets but it is not able to receive anything from its neighbours since they are not involved in communication.

Setting the broadcast probability to be very low, e.g. 10% in our implementation, leads to negligibly low network coding benefit. Another illustration of system randomness applied in our implementation is topology oriented. We generate random topology by defining probability of link existence between nodes and monitor network coding behavior under these circumstances. Results are similar to the case of random traffic pattern in the sense that network coding benefit is proportional

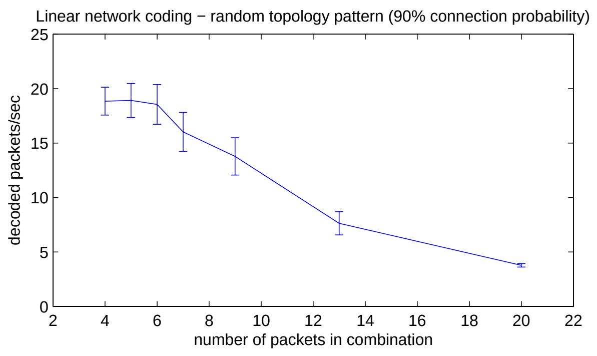

Fig. 16. Random topology pattern, mean number and standard deviation of decoded packets (90% probability of link existence between two nodes)

to the given link existence probability, and it degrades dramatically and can be disregarded for low probabilities due to network disconnections. Opposed to the random traffic pattern case, standard deviation of number of packets decoded is relatively small since there is no possibility for node to send neither to receive to/from no existing links as shown in Figure 16. In order to better understand the dependency of decoding benefit due to network connectivity, we extend out observations to larger networks and ”critical” connection probability between nodes. We choose 55% link existence probability to avoid cases of obvious network disconnections due to low probabiliy, e.g. 10%, and cases of high nodes connection probability, e.g. 90%, where circumstances are rather the same as in completelly connected networks. Decoding gain is not anymore proportional to connection probability as it was in the case of high connection probability.

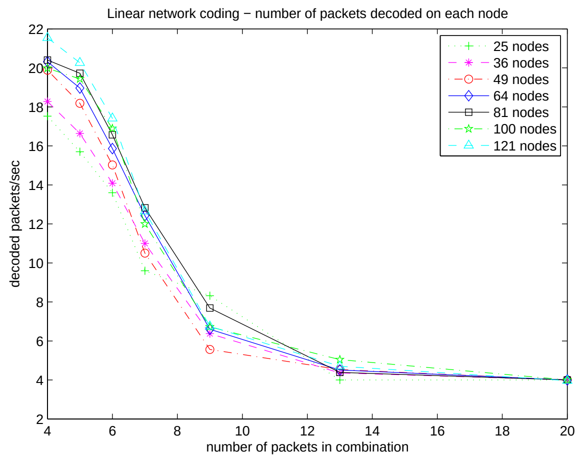

Fig. 17. Linear network coding. Grid networks built upon 25, 36, 49, 64, 81, 100 and 121 nodes with 100% connection probability between nodes

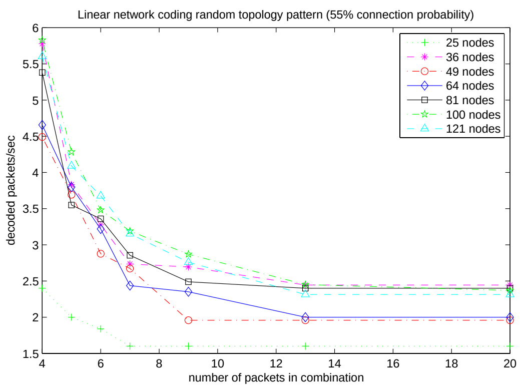

Fig. 18. Linear network coding - grid networks built upon 25, 36, 49, 64, 81, 100 and 121 nodes, with 55% connection probability between nodes

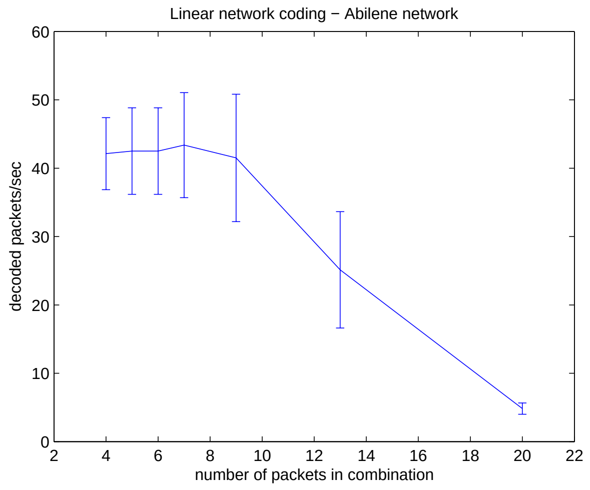

Fig. 20. Linear network coding - Abilene topology, mean number and standard deviation of packets decoded on each router

Results show that decoding curve has the same shape as for completly connected networks but the network coding gain is much lower due to network disconnections and nodes lacking sufficient information to perform better coding/decoding as shown in Figures 17 and 18.

B. Abilene network

As a final extension of our work we monitor behaviour of network coding applied on realistic topologies. We use high speed research network Abilene [1], [17] as a testcase.

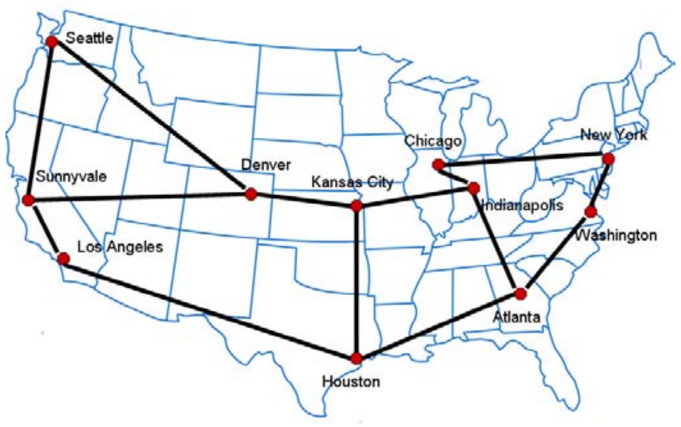

Fig. 19. Abilene backbone topology

Abilene is based on 10Gbps links between nodes, mainly serving to educational and research purposes. It represents advanced backbone network with the goal of supporting advanced network applications and evaluation of their performance before integration with widely used systems. Configuration files of Abilene routers are publicly available thus the exact topology settings of this network can be used for simulation and testing purposes.

Our topology consists of 12 nodes with link capacities and connections as described in Abilene backbone specifications. Network is shown in Figure 19, and we are using network coding in communications. Our aim is to use the same traffic generation parameters as applied in topologies that we have been already tested. Thus, the evaluation goal is to find out the impact of Abilene topology on network coding performance. Results show network coding benefit followed by large deviations in number of packets additionally decoded as shown in Figure 20. Relatively poor decoding gain is related to traffic generation manner used.

V. Conclusions

We have presented the main idea behind network coding mechanism while focusing our evaluation work on using XOR and linear network coding. Our results show that combining numerous data units in the way that its designated destination is able to recover it, while saving the number of data transmissions at the same time is not straightforward task. Gains of applying such a mechanism are relying on various input parameters in terms of channel properties, traffic patterns and related network topology. In addition, possible latency and complexity issues must be taken into account while using it under different scenarios. Finally, apart of parameters and topologies that have been analyzed in this work, there are many potentially interesting network settings to be investigated in the future. As an illustration of possible extensions to the current work similar evaluation could be done for wireless network case. Additionally, network coding employed under different traffic generators, larger networks and other existing realistic networks can be tested in order to get more complete picture of network coding manners and its applicability in specific use cases.

Acknowledgements

This work was in part financially supported by RWTH Aachen University through UMIC research centre and European Union (PSIRP project).

REFERENCES

[1] "The Abilene Research Network," http://www.cs.utexas.edu/~yzhang/research/AbileneTM, [visited on 20.04.2009.].

[2] R. Ahlswede, N. Cai, S.-Y.R. Li and R.W. Yeung, "Network Information Flow," *IEEE-IT*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[3] S. Katti, D. Katabi, W. Hu, H. Rahul, M. Médard, "The Importance of Being Opportunistic: Practical Network Coding for Wireless Environments," *in Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[4] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, J. Crowcroft, "XOR in The Air: Practical Wireless Network Coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, June 2008.

[5] T. R. Henderson, M. Lacage, G. F. Riley, "Network Simulations with the ns-3 Simulator," in *ACM SIGCOMM08*, Seattle,Washington,USA, Aug. 2008, demo abstract.

[6] B. Gajic, J. Riihijärvi and P. Mähönen, "Performance Evaluation of Network Coding in Middle-Sized Networks," submitted to PIMRC 2009.

[7] P. A. Chou, Y. Wu, K. Jain, "Practical Network Coding," *in Proc. 41st Annual Allerton Conference on Communication, Control, and Computing*, Oct. 2003.

[8] C. Fragouli, J. Le Boudec, J. Widmer, "Network Coding: An Instant Primer," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 63–68, Jan. 2006.

[9] R. Koetter and M. Médard, "An Algebraic Approach to Network Coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[10] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, B. Leong, "A Random Linear Network Coding Approach to Multicast," *IEEE/ACM Transactions on Networking*, vol. 52, no. 10, pp. 782–795, Oct. 2006.

[11] D. S. Lun, M. Médard, R. Koetter, M. Effros, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, no. 1, pp. 3–20, March 2008.

[12] S.-Y. R. Li, R. W. Yeung, N. Cai, "Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 49, pp. 371–381, 2003.

[13] C. Fragouli, A. Markopoulou, "A network coding approach to overlay network monitoring," *in Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2005.

[14] J. Barros, R. A. Costa, D. Munaretto, J. Widmer, "Effective Delay Control in Online Network Coding," in *In Proc. of IEEE INFOCOMM*, Rio de Janeiro, Brazil, April 2009.

[15] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, J. Barros, "Network Coding Meets TCP," in *in Proc. of IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[16] C. P. R. Braden, D. Borman, " Computing the Internet Checksum," RFC 1071, 1988. [Online]. Available: http://www.ietf.org/rfc/rfc1071.txt

[17] R. Sherwood, A. Bender, N. Spring, "Discarte: a disjunctive internet cartographer," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 303–314, Oct. 2008.