Network Coding forWireless Mesh Networks: A Case Study

Anwar Al Hamra, Chadi Barakat, Thierry Turletti

Network Coding for Wireless Mesh Networks: A Case Study

Anwar Al Hamra
Department of Informatics
University of Oslo - Norway
anwarah@ifi.uio.no

Chadi Barakat, Thierry Turletti
INRIA - Planète research group
Sophia Antipolis - France
{cbarakat, turletti}@sophia.inria.fr

Abstract

Network coding is a new transmission paradigm that proved its strength in optimizing the usage of network resources. In this paper, we evaluate the gain from using network coding for file sharing applications running on top of wireless mesh networks. With extensive simulations carried out on a simulator we developed specifically for this study, we confirm that network coding can improve the performance of the file sharing application, but not as in wired networks. The main reason is that nodes over wireless cannot listen to different neighbors simultaneously. Nevertheless, one can get more from network coding if the information transmission is made more diverse inside the network. We support this argument by varying the loss rate over wireless links and adding more sources.

1 Introduction

A few years ago, an elegant transmission paradigm, called network coding, has been proposed by Ahlswede et al. [2]. The key idea is that, one can approach the broadcast capacity of the network by allowing intermediate nodes inside the network to code and decode the information carried by the different flows. Since then, network coding's popularity is increasing and many research papers have appeared on the subject [12, 3, 7, 8, 4]. Most of these papers focus on the multicast case where all receivers are interested in the same information. As a consequence, only the coding task has to be done inside the network while the decoding process is done at the receivers in order to reconstruct the original information. This is relatively simple to implement compared to the unicast case which requires an intelligent decoding inside the network to be profitable [6, 9].

Two main scenarios have profited from the power of network coding: file sharing applications [5] and wireless mesh networks [8, 4]. File sharing is the application allowing a certain number of users to collaborate together in order to share a certain content. A wireless mesh network is a one where any two hosts can communicate directly over wireless, if their transmission powers allow, otherwise indirectly via the other hosts. Network coding has proved its capacity to improve the resource utilization in both scenarios, separately. In this work we try to answer the question of how network coding performs when these two scenarios are combined together. We believe that file sharing over wireless mesh networks is an emerging domain with lot of potential applications. Exchanging files, updating databases and pushing new software releases are typical examples of applications.

1.1 Data Encoding in Network Coding

We describe briefly a practical approach for network coding that we borrow from the work in [5]. This approach splits the file into multiple pieces called chunks. Denote by $|\mathcal{C}|$ the number of chunks. When a node has chunks that are of interest to another node, it generates and sends a linear combination of all the chunks in its possession (similarly to XORing multiple chunks). Let us start with a node A that holds the entire file, i.e., the $|\mathcal{C}|$ chunks. When A decides to transmit some information to neighbor B, it first picks up at random $|\mathcal{C}|$ coefficients $k_1, \ldots, k_{|\mathcal{C}|}$. Then, it multiplies chunk C_i with K_i and finally adds all resulting chunks

together, i.e., $C' = C_1 \ldots K_1 \oplus \ldots \oplus C_{|\mathcal{C}|} \cdot K_{|\mathcal{C}|}$. In this case, node A should send to node B not only the combination C', but also the set of coefficients $k_1, \ldots, k_{|\mathcal{C}|}$. The source encoding process is repeated at intermediate nodes on the chunks received coded by upstream nodes. After a node receives enough linearly independent combinations of chunks, exactly $|\mathcal{C}|$ combinations, it can reconstruct the original file. Note that, nodes need not wait to retrieve the whole file to encode the data. Suppose that node B receives another combinations C" from node C. Node B can then encode C' and C" and send the resulting combination $C' \cdot K'_1 \oplus C" \cdot K'_2$ to interested nodes.

One would wonder why node A does need to generate random coefficients: *Why does n't node A simply transmit a XOR of all chunks, i.e.,* $C' = C_1 \oplus \ldots \oplus C_{|\mathcal{C}|}$. Actually, a simple XOR combination is a very bad choice for the following reason. If node A holds $|\mathcal{C}|$ chunks, it should be able to generate $|\mathcal{C}|$ disjoint combinations, which can not be done through a simple XOR. At the same time, generating only random coefficients is not enough to obtain disjoint combinations. Instead, the set of coefficients selected for each transmission should be new. To ensure so, the field from which we select the random coefficients must be large enough. If we choose a field size of 2^{16} (i.e., $0 < K_i < 2^{16}$), we can ensure that the probability that two nodes generate two identical sets of coefficients is extremely low.

What we have not yet mentioned is that, upon receiving an announcement for a combination, a node checks out whether it is a useful one or not. We say that the combination is useful only if it is linearly independent of the other combinations held at the node. One way to check dependency is by performing the Gaussian algorithm [1]. If the combination is new, then the node asks for it; otherwise, the combination's announcement is discarded.

1.2 Network Coding and File Sharing

The semantics of network coding and its appealing features make it a good candidate for improving the performance of file sharing applications. Indeed, instead of forwarding a piece of the file, a node forwards a coded version of all the pieces it has seen so far. This combined version can serve to all downstream nodes missing some of the individual pieces. Without network coding and in the absence of information on the status of all downstream nodes, the node is unable to forward the piece that satisfies everyone. The use of network coding should transform into a shorter time to share the file and a better utilization of network resources.

The idea of network coding for file sharing has been recently introduced by Gkantsidis et al. [5]. In their work, the authors propose to use network coding for the distribution of large content in P2P (Peer-to-Peer) networks. The authors show that, in some scenarios, network coding can reduce the distribution time of the content by up to three times as compared to no coding at all. The explanation is simple. In P2P networks, a node needs to decide what chunks to retrieve from each neighbor. Clearly, retrieving globally rarest chunks is more efficient. Like this, the node can significantly contribute to the distribution process as a lot of nodes will be interested in these rarest chunks. However, to know what chunks are globally rarest, nodes need to be synchronized, which is practically infeasible in large networks. In this context, network coding removes the need for such a synchronization. A node retrieves a coded version of all the data maintained by its neighbor. As a result, the notion of rarest chunks is completely eliminated and the content distribution process is greatly improved.

1.3 Network Coding and Wireless Mesh Networks

The broadcast feature of wireless networks makes them good candidates for the application of network coding. When a node transmits a piece of information to one of its neighbors, this information is heard by the other neighbors as well. Thus, a node whose neighbors are interested in different pieces can transmit a coded version of all these pieces it possesses, which will profit to all the neighbors and save the wireless resources.

In this work, we want to check the gain from using network coding in file sharing applications over wireless mesh networks. The study in [5] only applies to wired networks. Its extension to wireless networks is not straightforward for many reasons. In P2P wired networks, connections between peers are unicast and a

node can communicate with any other node in the system by just knowing its IP address. Moreover, a node can retrieve (respectively upload) different parts of the file from (respectively to) different nodes at the same time.

In wireless mesh networks, the situation is different. First, when a node transmits a piece of the file to a neighbor, the piece is heard by the other nodes in its range, which can avoid further transmissions. This is one positive point in favor of wireless networks. The negative points are diverse, however. A wireless node with one single antenna (which is commonly the case) is not able to receive from more than one neighbor at the same time. Also, wireless nodes in some neighborhood contend to the medium independently of the piece of data they have. This may prevent the most relevant pieces to be transmitted first. In addition, nodes are not free in their choice of neighbors. Instead, neighbors are imposed by the physical layer, which poses a tight constraint on the cooperation among nodes in the network.

When we thought about network coding for file sharing over wireless mesh networks, we had the following scenario in mind. Assume three nodes, A, B, and C, where node A holds chunks $\{C_1, C_2\}$, node B holds chunk C_1, and node C holds chunk C_2. If network coding is not used, node A needs first to transmit chunk C_2 to node B and then, chunk C_1 to node C. In contrast, if network coding is used, node A transmits one single combination $C_1 \cdot K_1' \oplus C_2 \cdot K_2'$, and both nodes B and C can retrieve the chunk they miss.
Despite its simplicity, we can learn two lessons from this example. The first one is that, network coding reduces both the number of chunks transmitted and the number of useless chunks received by the different nodes. Reducing the number of transmitted chunks not only saves bandwidth but it also speeds up the distribution process. In addition, by reducing the number of useless chunks received, we reduce the battery consumption at the different nodes. The second lesson is that, to be efficient, network coding requires neighbors to have disjoint set of chunks. We can easily verify that, if nodes B and C have both the same chunk, say C_1, network coding will not bring any benefit to the system.

Diversity of information in the network is then very helpful for network coding. When coding is done at the packet level independently of the application, the diversity improves if a large number of flows and routes are multiplexed together [8]. In our context where coding is done intra-flow, diversity can be ensured by other means. For example, one can introduce diversity by multiplying the number of sources. We evaluate this suggestion later. The topology of the network decides on diversity and determines the speed at which the information spreads. For example, a ring or a cascade of rings leads to more diversity than a line or a strip. Chunk losses over wireless and the fact that some nodes can sleep from time to time to save resources, also increase the diversity of content in the network and create situations in favor of network coding. We control diversity in our simulations by the means of losses that can model both the bad conditions of the wireless and the intermittent sleep of nodes.

1.4 Methodology

We assume the existence of N fixed wireless nodes having a well defined transmission radius that want to share a file of size $|\mathcal{C}|$ chunks. At the beginning, few nodes possess the entire file, which we call the sources. In this context, our goal in this paper is to derive the conditions under which network coding is beneficial. To this end, we develop a simulator that calculates the time required to share the file as well as the number of chunks transmitted and those unnecessarily received. We conduct extensive simulations where we identify the system parameters that influence the performance of network coding. These parameters include the cooperation strategy between nodes, the number and placement of the sources of the file, the loss rate in the network, the number of nodes, and the number of chunks. We believe that our results and discussions form a step towards the design of efficient content distribution approaches in wireless networks.

1.5 Organization of this Paper

The rest of this paper is organized as follows. Section 2 introduces the cooperation strategies between nodes that we evaluate. In Section 3, we give a short description of our simulator. Section 4 provides simulation results and we conclude the paper with a discussion in Section 5.

2 Cooperation Strategies between Nodes

We consider two extreme and opposite classes of cooperation strategies between nodes. In the first one, which is the worst indeed, we assume a very primitive cooperation based on flooding. This class includes two strategies namely, *BF* and *BF-NC*, which account for the cases with and without network coding.

- *BF* (*Blind-Forwarding*).

  When a node receives a new chunk and after getting access to the medium, it blindly forwards the chunk to its neighbors even if none is interested. Given that neighbors share the same wireless medium, a node may receive multiple chunks before it could perform one single transmission. In this case, and whenever the medium is free, the node schedules the chunks that have not yet been transmitted in the same order of receiption, i.e., first received first transmitted. Under this strategy, every node transmits as many chunks as it receives.

- *BF-NC* (*Blind-Forwarding* with network coding). In this strategy, when a node receives a new combination, it competes to get access to the medium. When it succeeds, it generates and transmits a combination of all information in its possession and forwards the result to its neighbors. Each receiption of a new combination leads to a new transmission and therefore, every node transmits as many chunks as it receives.

The second class that we consider is completely the opposite, but it also includes two strategies, *SF* and *SF-NC*. In this class, we assume that every node maintains a table that offers a complete knowledge about the list of chunks at each of its neighbors. In practice, building and maintaining these tables can be done through update messages exchanged regularly between neighbors. How to build and maintain these tables is however out of the scope of this paper.

- *SF* (*Selective-Forwarding*). Every node checks continually its table. If it finds a chunk that is of interest to at least one of its neighbors, the node tries to transmit that chunk. In case there are many chunks, the rarest one is selected. Finally, if there are many chunks with the same priority, one is chosen at random.

- *SF-NC* (*Selective-Forwarding* with network coding). As before, every node checks continually its table looking for neighbors interested in the information it holds. If at least one is found, the node generates and transmits a combination of all what it posses. Notice that the problem of identifying rarest chunks does not exist in this case.

It is clear that *SF* and *SF-NC* achieve a more efficient bandwidth and energy utilization as compared to *BF* and *BF-NC*, but, at the expense of a more complex implementation. Our goal in this paper is not to prove this obvious conclusion. Instead, we aim at understanding how network coding performs under different cooperation strategies and these four ones are good candidates. They are intuitive, simple and thus, permit to get new insights while keeping clear the analysis. Moreover, these are extreme strategies and pave the way to derive many others by combining them.

3 Simulation Methodology

To evaluate our four strategies, we developed a C++ simulator[1] that allows us to observe step-by-step the distribution of the file among all nodes in the system. Before going into the implementation details, we first define the parameters that we will use in our analysis.

3.1 Notations

$|\mathcal{C}|$ is the number of chunks in the file to distribute. b represents the rate at which nodes transmit their chunks over the wireless medium. We take as *one unit of time*, the time needed to download the entire file at rate b. *One round* is the time needed to download one single chunk at rate b. It follows that, for a file of $|\mathcal{C}|$ chunks of equal size, one round is equivalent to $\frac{1}{|\mathcal{C}|}$ unit of time.
N stands for the number of nodes while R represents their transmission radius. We denote by S the area where the N nodes are deployed. S_n refers to the number of sources in the system while S_p refers to their placements. For instance, ($S_n = 1$, $S_p = Random$)

[1]The simulator is available for public access at the following address: http://www-sop.inria.fr/planete/Software/NCWM/

means that there is one single source that is placed at random within the area S. Finally, the parameter ℓ stands for the loss rate in the network. $\ell = 20$ means that a node receives the chunks from its neighbors with a probability of 20%. Notice that the chunk loss rate is a function of many factors such as, the packet loss rate, the sleep mode of nodes, and the way lost packets are treated by the MAC layer.

3.2 Implementation Details

Our simulator makes an abstraction of the MAC layer by assuming an ideal MAC protocol without collisions and that gives contending nodes the same probability to access the medium. We are of course aware that this assumption does not hold true in practice. One can see our work as a proof-of-concept about the benefit of network coding for wireless mesh networks. In addition, this assumption allows us to draw broad conclusions and to provide new insights without being limited to a specific MAC protocol. As a next step, we intend to implement network coding on top of a variety of MAC protocols including 802.11.

Our simulator splits the time space into rounds each of $\frac{1}{|\mathcal{C}|}$ unit of time. A node can transmit only at the beginning of a round. During each round, we go through the following four steps. In the first step, we identify the candidate transmitters, i.e., nodes that have information to send. These candidate transmitters are then placed in a list, called the *Candidates-List*. Given that we assume an ideal MAC protocol without collisions, only a part of these candidate transmitters can perform a transmission. The selection of the transmitters is done during the second step as follows. We pick up at random one candidate transmitter, let us say node A, and move it from the *Candidates-List* to a new list that we call *Transmitters-List*. Then, to ensure that there are no collisions, we prohibit all A's neighbors from transmitting during the current round. As a consequence, A's neighbors are deleted from the *Candidates-List* (if there is any). We also delete from *Candidates-List* the neighbors of A's neighbors. The purpose is to avoid the situation where a node receives from more than one neighbor at once (i.e., the hidden terminal situation). Then, the *Candidates-List* is accessed again and a new node is moved to the *Transmitters-List* and so on. This process continues until the *Candidates-List* becomes empty. This selection algorithm of the transmitters is simple and fair at once. We can easily verify that, the probability of selecting a transmitter takes into account the neighbors and neighbors of neighbors that are willing to transmit and give them the same chance to access the medium. In the third step, we scan the *Transmitters-List* and all selected transmitters send one chunk each. In the fourth and final step, we update the list of chunks at all nodes. These four steps are performed recursively until all nodes receive the entire file.

4 Performance Evaluation

The goal of this section is to highlight the parameters that guide the benefit of network coding for file sharing in wireless mesh networks. We aim at deriving the scenarios under which network coding improves the system performance. We are also curious of exploring the conditions that limit the gain from network coding.

4.1 Parameters Used in the Simulations

Our system has a quite few parameters, but we present results for a limited subset of parameter values that can provide new insights. For the rest of the results, we will vary the number of chunks $|\mathcal{C}| = \{10, 50, 100\}$, the number of nodes $N = \{10, 25, 50, 75, 100\}$, the placement of the source, and the loss rate (in percent) in the network $\ell = \{0, 20, 50, 80\}$.

Other parameters are kept unchanged and chosen as follows. The transmission radius of all nodes is set to $R = 2$ units. However, our work and results can be easily extended to the case of dynamic and adaptive radius. The N nodes are randomly deployed within a square of area S. As a consequence, the values of S, N, and R have to be chosen carefully so that the density of nodes is high enough to ensure total connectivity. These values must satisfy the following inequality as showed in a previous study by Philips et al. [10]: $N \geq \frac{10 \cdot S}{\pi \cdot R^2}$. In this paper, we choose $N = \frac{10 \cdot S}{\pi \cdot R^2}$. For instance, when $R = 2$ and $S = 11^2 = 121$, the number of nodes should be $N = 96$. As a result, when changing N, we change the area of deployment in accordance while keeping

the radius constant, i.e., the nodes density remains constant.

4.2 Metrics

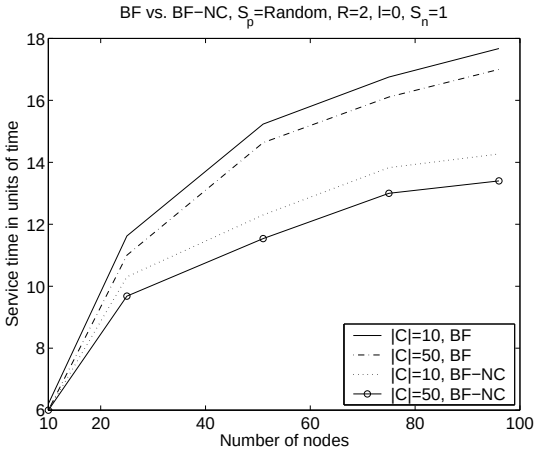
In our evaluation, we consider several performance metrics including:

- *Service time.* The time needed to distribute the whole file to all nodes in the system. We compute this metric in terms of units of time. In general, a short service time is desirable.
- *Emitted chunks.* The number of transmitted chunks by all nodes during the entire simulation.
- *Useless chunks.* The number of useless chunks received by all nodes and during the whole simulation.

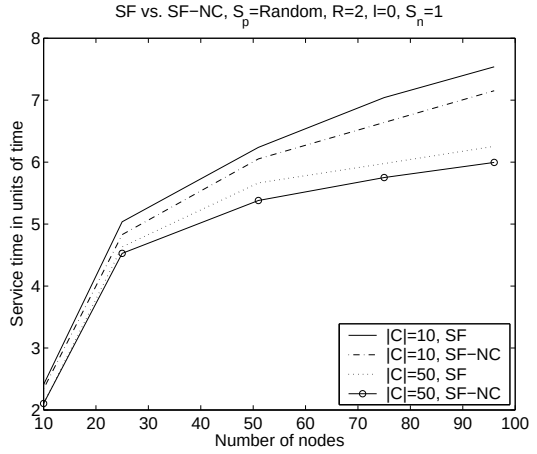
Achieving a low number of emitted/useless chunks is essential to save energy and bandwidth resources. Yet, these two metrics are meaningful only under *SF* and *SF-NC*. Actually, in *BF* and *BF-NC*, these metrics provide no new insights at all and their values are known in advance. For instance, under the scenario where there is no loss, every node transmits $|\mathcal{C}|$ chunks. In addition, every node receives $|\mathcal{C}|$ chunks from each neighbor.
Note that, we also evaluated the evolution of chunks in the system, how many new chunks (or combinations) are received each $\frac{1}{|\mathcal{C}|}$ unit of time. Yet, these metrics are ignored because they present no new insights as compared to the above three ones. Our results are averaged over multiple runs. We keep running simulations until the 90% confidence interval is within 3% of the average values.

4.3 Preliminary Hints and Observations

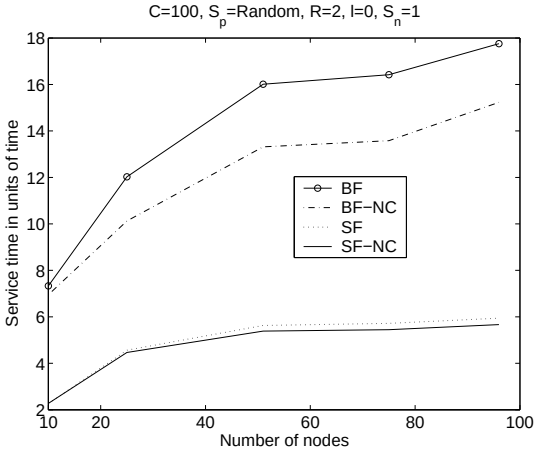
We start our performance evaluation with a simple scenario where we assume one single source that is placed at random within the area S. We also assume that there is no loss at all. Under this scenario, Figure 1 plots the service time against the number of nodes and for a variety of number of chunks. From this figure, we can make many interesting observations. Mainly:



(a) *BF* vs. *BF-NC*, $|\mathcal{C}| = \{10, 50\}$



(b) *SF* vs. *SF-NC*, $|\mathcal{C}| = \{10, 50\}$



(c) *BF*, *BF-NC* vs. *SF*, *SF-NC*, $|\mathcal{C}| = 100$

Figure 1. Service time against number of nodes for *BF*, *BF-NC*, *SF*, and *SF-NC*. The number of chunks is set to $|\mathcal{C}| = \{10, 50, 100\}$. There is one single source that is placed at random within the deployment area S. The loss rate is set to $\ell = 0$.

a) Network coding speeds up the distribution of the content. Under some conditions, the service time can be reduced by 25%.

b) Network coding requires a moderate to large number of nodes to be profitable to the system. This is a logic result since network coding becomes more efficient when many neighbors benefit from the same transmission. When the number of nodes in the system gets larger, nodes would probably have more neighbors from which to get the chunks and the information in the network would be diversified. Consequently, the potential gain from each transmission would also increase.

c) Increasing the number of chunks improves the performance of all schemes. Indeed, when we divide the file into multiple chunks, a node can start serving a chunk as soon as it finishes downloading it instead of waiting to download the whole file. As a result, the larger the number of chunks, the faster the nodes are engaged into the distribution process and the better the system performance.

d) The benefit of network coding depends dramatically on the cooperation strategy between nodes. As we can easily observe in Figure 1, *BF-NC* performs much better than *BF* while *SF* and *SF-NC* show similar behavior.
To better understand this last point, let us revisit the design principles of our cooperation strategies. With *BF*, a node forwards the chunks it holds even if none of its neighbors is interested. The main drawback is that a lot of useless chunks are transmitted and nodes progress slowly in the download process. In contrast, under *SF*, a node only sends chunks that are of interest to its neighbors. As a result, there are no useless transmissions and nodes progress faster as compared to *BF*.

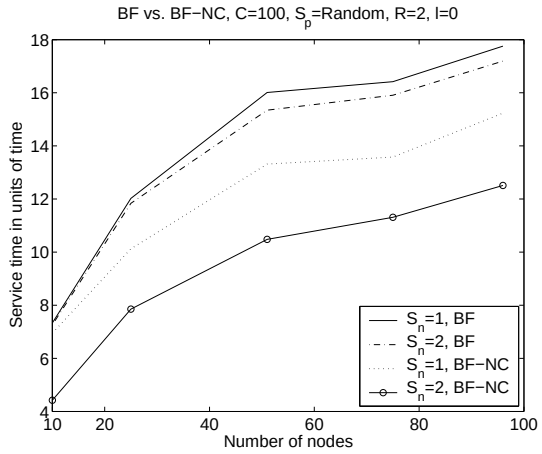
When network coding is employed under *BF*, it helps reducing the number of useless chunks transmitted. On the other hand, with *SF*, network coding helps increasing the number of neighbors interested in each transmission. We can conclude that the reason that makes network coding beneficial is not the same in both, *BF* and *SF*.

4.4 Getting More from Network Coding

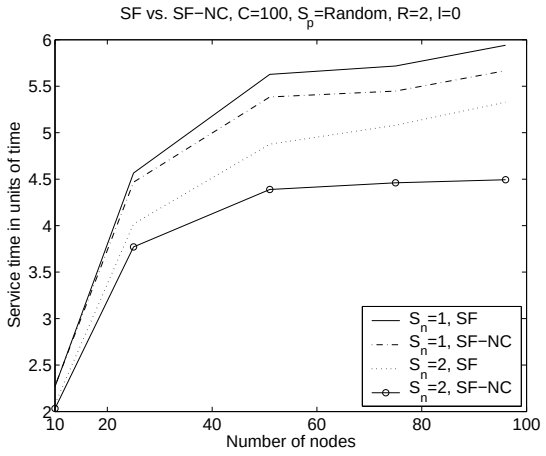
The low benefit of network coding that we observed under *SF* is mainly because, it is not very common for a node to have its neighbors interested in different chunks. This result was somehow expected because of the shared nature property of wireless that we mentioned in the introduction. This property does not help creating diversity of information in the network, which is essential for network coding under cooperation strategies like *SF*. By diversity, we mean that neighbors should have different sets of chunks. Many conditions can lead to this diversity including, having multiple sources and having non zero loss rates. In this section, we will consider these two scenarios and discuss how they significantly influence the performance of network coding.

4.4.1 Allowing Multiple Sources

In Figure 2, we graph the service time for all schemes as a function of the number of nodes and for $|\mathcal{C}| = 100$. This figure confirms our intuition. Having multiple sources makes the information in the network more diverse, which greatly benefits network coding. In contrast to the case with one single source, the outperformance of network coding is not limited to *BF*, but is also clear under the *SF* strategy. Figure 2 shows that, when adding a second source, the performance of *BF* doubles while that of *SF-NC* becomes up to three times better. For instance, when $N = 100$, and compared to *SF*, *SF-NC* reduces the service time by about 6% when there is one source while this reduction is about 15% when there are two sources. Moreover, the benefit of network coding appears at many levels including the number of useless chunks received and those emitted by the different nodes in the system (see Figures 3 and 4). For more clarity, in figure 4 we give the ratio of the number of emitted chunks in *SF-NC* to *SF* vs. the number of nodes. The result in Figures 3 and 4 is very important as achieveing a lower number of emitted/received chunks turns out to less battery consumption, which is critical for some devices such as wireless sensors. Note that, we also studied the scenario with three sources and network coding showed even a better performance.



(a) *BF* vs. *BF-NC*



(b) *SF* vs. *SF-NC*

Figure 2. Service time against number of nodes for *BF*, *BF-NC*, *SF*, and *SF-NC*, and for a number of chunks $|\mathcal{C}| = 100$. We consider two values for the number of sources $S_n = \{1, 2\}$ while the loss rate is set to $\ell = 0$.

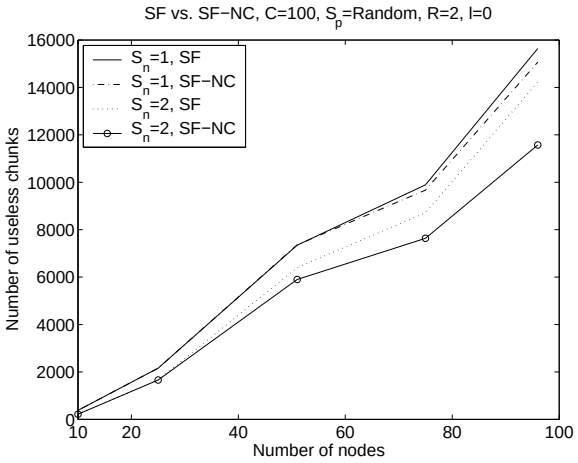


Figure 3. Number of useless chunks received vs. number of nodes for *SF* and *SF-NC*. The number of chunks is $|\mathcal{C}| = 100$ and the loss rate is $\ell = 0$. We consider two values for the number of sources $S_n = \{1, 2\}$.

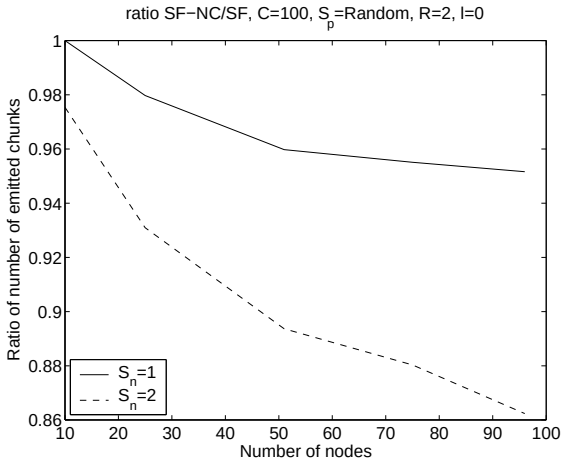


Figure 4. Ratio of the number of chunks emitted in *SF-NC* to *SF*. The number of chunks is $|\mathcal{C}| = 100$ and the loss rate is $\ell = 0$. We consider two values for the number of sources $S_n = \{1, 2\}$.

Interaction Between Network Coding and Placement of the Sources: Figures 2, 3, and 4, assume that nodes are placed at random within the area S. We argue that one could get more from network coding by carefully placing the sources in the network. To support this claim, we plot in Figure 5 the service time vs. number of nodes for *SF-NC*. In this figure, we take

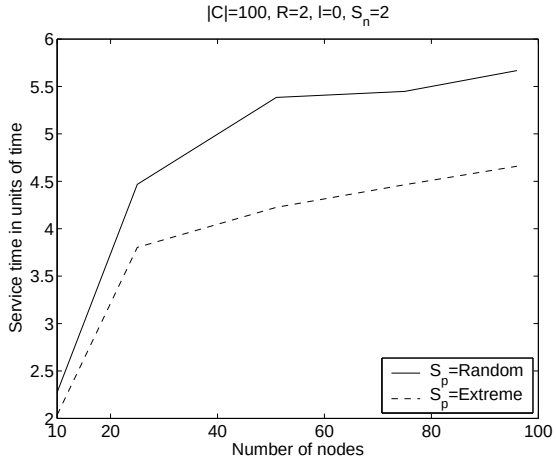


Figure 5. Service time for *SF-NC* vs. number of nodes with two sources at two different placements $S_p = \{Random, Extreme\}$. We choose a number of chunks $|\mathcal{C}| = 100$ and a loss rate $\ell = 0$.

$|\mathcal{C}| = 100$ and two sources that are either placed at random or at the extremities. By extremities we mean the two opposite corners of S. More precisely, the first source has coordinates $(0, 0)$ while the second one has (X, Y), where X and Y are the length and the width of S.

Figure 5 shows clearly that the placement of the sources is a critical factor that dramatically impacts the performance of network coding. Compared to Figure 2, we can observe that this particular placement of the sources improves significantly the gain. This result is extremely interesting as it paves the way for a mathematical work to derive the optimal placement as well as the optimal number of sources in the network.

4.4.2 Network Coding Performs better in Lossy Environments

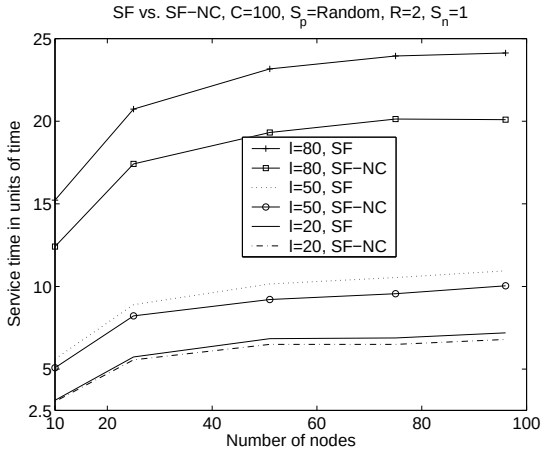
In addition to having multiple sources, chunks' losses can also create diversity of information at the different nodes, which makes network coding more efficient. Loosing chunks can be due to bad wireless conditions or a consequence of intermittent sleep of nodes. Sleep mode is often used in scenarios where battery energy is a scares resource.

In this paper, we use a very simplified model for losses. Every node receives the chunks from its neighbors with a probability $1 - \frac{\ell}{100}$. We consider three values of loss rate $\ell = \{20, 50, 80\}$. Under this scenario, we evaluate the service time and the number of emitted chunks for *SF* and *SF-NC* as shown in Figure 6. Our conclusions also apply to *BF* and *BF-NC*. As we can point out from the figure, network coding is efficient under relatively high loss rates, i.e., $\ell \geq 50\%$. When the loss rate is equal to $\ell = 20$, *SF* and *SF-NC* perform almost equally. The explanation is simple. The network that we consider here is quite dense and each node has, on average, about **8** neighbors. A loss rate of $\ell = 20$ means that, each chunk is on average delivered to 80% of neighbors, which gives 6 out of 8. As a result, even if a node looses a chunk, there are other neighbors that receive it and would not take long to forward it during the next rounds. Thus, the node can rapidly retrieve what it missed from common neighbors and thus, the content at the different nodes remains homogeneous.

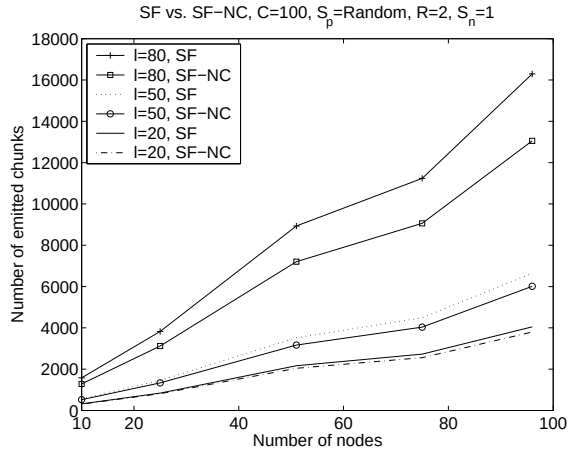
What is also encouraging in the result of Figure 6 is that, network coding can be employed as an efficient solution for content distribution under very bad environment conditions. These bad conditions can be caused by the wireless physical layer or also by a poor MAC protocol that produces a lot of collisions.

5 Discussions and Conclusions

In this paper we studied the use of network coding for file sharing over wireless mesh networks. Through extensive simulations, we investigated how this transmission paradigm behaves under a variety of scenarios. The importance of our study is that we identified the main parameters that influence the performance of network coding in wireless environment. These parameters include the number of nodes, the number of



(a) Service time



(b) Emitted chunks

Figure 6. Service time and number of emitted chunks against number of nodes for *SF* and *SF-NC*. We set the number of chunks $|\mathcal{C}| = 100$ and the number of sources $S_n = 1$. We consider several values of the loss rate $\ell = \{20, 50, 80\}$.

chunks, the number and placement of the sources, the cooperation strategy between nodes, and also the loss rate in the network. We showed how these parameters interact with each other and influence the behavior of network coding.

A key result of our work is that, employing network coding should take into account several environment conditions. For instance, when using a cooperation strategy such as *SF* and under good wireless conditions with no losses, there is no need for network coding. In contrast, if the cooperation strategies between nodes is somehow primitive (e.g., to simplify implementation issues), or if the wireless connections experience high loss rates, network coding can be a very good candidate to significantly improve the system performance. Finally, the existence of multiple sources for the content is in favor of the deployment of network coding.

5.1 Overhead of Network Coding

The benefit that one can achieve with network coding comes at the expense of more battery and CPU time consumption for encoding and decoding the information. In our case, the encoding process is very easy because it is only about generating a linear combination of all information together available at each node. In contrast, the decoding task is more complex because it requires to run a test to check independency. The complexity of this test is a function of the number of chunks $O(|\mathcal{C}|)$. Nevertheless, the tendency nowadays is to design devices that consume much less energy battery for computation tasks. However, the effort spent by nodes for transmitting and receiving the chunks will remain a primary source of energy consumption. A comparison of the cost of computation to communication in future platforms by Pottie et al. [11] reveals that 3000 instructions can be executed for the same cost as the transmission of one bit over 100m. In this context, and under some scenarios, network coding is a suitable solution as it can significantly reduce the number of useless chunks received as well as the number of transmissions performed by the different nodes.

In addition to the computing overhead, network coding requires nodes to transmit not only the coded chunk, but also the list of the coefficients that have been used in the encoding process. Yet, this bandwidth

overhead is negligible as compared to the size of the chunk in transmission. It can be computed as $K_s \cdot |\mathcal{C}|$ Bytes, where K_s is the size of each coefficient. For instance, $C_s = 2$ Bytes when the coefficients are selected from a field of size 2^{16}. Hence, if we consider a file that includes 50 chunks with each of 10 KBytes, the bandwidth overhead is less than 1%.

5.2 Open Research Issues

In our work, we made several assumptions to simplify the analysis. On the one hand, we assumed an ideal MAC protocol without collisions. On the other hand, we split the time space into rounds and nodes were able to transmit only at the beginning of each round, i.e., using global synchronization. We are aware that these assumptions do not hold true in practice. Yet, the goal of this study was to provide new hints and observations about the power of network coding for file sharing in wireless environments. The above assumptions allowed us to give new insights and draw broad conclusions without being limited to a specific MAC protocol. We believe that this work sheds a new light on new solutions for content distribution in the area of wireless mesh networks.

Future work can progress along many avenues. We are planning to implement and evaluate network coding on top of a variety of MAC protocols including 802.11. Within the cross-layer optimization philosophy, we could think of designing a MAC protocol that contributes in creating information diversity in the network and thus, making network coding more powerful. What would also be interesting is to derive the optimal number of sources as well as their optimal placement for a given deployment of nodes.

References

[1] Gaussian elimination. Wikipedia the free encyclopedia, November 2005. http://en.wikipedia.org/wiki/Gaussian_elimination.

[2] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network information flow. *In Proc. of IEEE Transactions on Information Theory*, July 2000.

[3] P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proc. of Allerton*, Monticello, IL, USA, September 2003.

[4] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun., M. Medard, and N. Ratnakar. Network coding for wireless applications: A brief tutorial. In *In proc of IWWAN*, London, UK, May 2005.

[5] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proc. of INFOCOM*, Miami, USA, March 2005.

[6] T. Ho and R. Koetter. Online incremental network coding for multiple unicasts. In *Proc. of DIMACS Working Group on Network Coding*, Piscataway, NJ, USA, January 2005.

[7] K. Jain, L. Lovasz, and P. Chou. Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding. Technical report, Microsoft Research, December 2004.

[8] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard. The importance of being opportunistic: Practical network coding for wireless environments. In *Proc. of Allerton*, Monticello, IL, USA, September 2005.

[9] Z. Li and B. Li. Network coding: The case for multiple unicast sessions. In *Proc. of Allerton*, Monticello, IL, USA, September 2004.

[10] T. K. Philips, S. S. Panwar, and A. N. Tantawi. Connectivity properties of a packet radio network model. *In Proc. IEEE Transactions on Information Theory*, September 1989.

[11] G. Pottie and W. Kaiser. Wireless integrated network sensors. *In Proc. of Communications of the ACM*, May 2000.

[12] Y. Zhu, B. Li, and J. Guo. Multicast with network coding in application-layer overlay networks. *In Proc. of JSAC*, January 2004.