DYGES: A network-aware Generation-Based Network Coding for multicast flows

Youghourta Benfattoum, Steven Martin, Khaldoun Al Agha
Laboratoire de Recherche en Informatique,
Bât 650, Université Paris-Sud, CNRS
Email: {youg, smartin, alagha} @lri.fr

***Abstract*—Most of the works on Generation-Based Network Coding (GBNC) consider a fixed generation size. A large generation size maximizes the Network Coding benefits but leads to a long delay while a small generation size reduces the delay but decreases the throughput. This paper presents the DYnamic GEneration Size (DYGES) approach. Our network-aware method adjusts the generation size according to the network variations (network size, congestion, losses) for multicast flows to keep the delay steady. Our goal is to guarantee a Quality of Service (QoS) in terms of delay. The simulation results show the accuracy of DYGES.**

I. INTRODUCTION

Introduced in 2000 [1], Network Coding is a recent technique that mixes the flows in order to increase the network performance. It was proved in [2] that for multicast traffic, linear codes are sufficient to achieve the maximum capacity bounds. Then, the authors of [3] proposed to solve the Network Coding problem in an algebraic way.

However, the coding/decoding scheme has to be agreed upon beforehand and needs full network topology. Therefore, the Linear Random Network Coding was proposed in [4] and proved to be efficient. This randomized coding guarantees robust distributed transmission and makes Network Coding easier to implement in a real network. COPE [5] is the first architecture for Network Coding in a wireless mesh network. It uses an opportunistic coding to choose the packets to XOR in order to maximize the number of neighbors that can decode them. Various approaches such as BFLY [6] and DCAR [7] propose to apply COPE with a different scheme in order to still increase the throughput.

In this paper we use a Generation-Based Network Coding (GBNC) for multicast flows. Using this approach, the source divides the file to send into blocks called generations (also referred as batches[1]). When sending a packet, it combines only the packets belonging to the same generation using coefficients randomly and uniformly selected from a Galois Field while preventing all-zero coding coefficients. Due to the variations of the network characteristics, the delay for decoding a generation can vary dramatically. Our goal is to adjust the generation size to insure a steady decoding delay. A potential application of our work is to guarantee the QoS for video streaming and conferencing in either wireless or wired networks. In this context, the end-user application is required to display information within a certain delay. Moreover, video conferencing needs the display synchronization among the participants.

[1] In the remaining of this paper, we refer to batch and generation interchangeably.

This paper is structured as follows: The issue we deal with is explained in Section II. We review the previous works relevant to Network Coding and delay in Section III. A theoretical background about the Network Coding parameters is explained in Section IV. We describe our approach in Section V and show its accuracy by simulation in Section VI. We conclude in Section VII.

II. PROBLEM STATEMENT

In GBNC systems, the generation size is a crucial parameter, it affects both delay and throughput. A large generation size maximizes the Network Coding benefits such as increasing the throughput but leads to a long delay. As for a small generation size, it reduces the decoding delay but decreases the throughput. Section IV gives more details about the relationship among the generation size, throughput and delay.

Most of the works on GBNC consider a fixed generation size. The network state is not smooth, it is subject to variations such as the number of nodes, congestions and losses. A small (resp. large) generation size is likely the most appropriate when the network load is high (resp. low). However, for a given context, if the chosen generation size is not adequate, the network performance decreases. Using for instance, a large generation size when the network is high loaded increases significantly the decoding delay. Even without being critical systems, some applications need to have a reasonable bounded decoding delay to guarantee a QoS for the end user.

Having this in mind, we propose DYGES which adapts dynamically the generation size according to the network load for multicast flows. Its goal is to keep the delay steady around a given threshold while maximizing the throughput corresponding to that delay. The next section gives a state of the art related to our approach.

III. RELATED WORK

The idea of GBNC was first introduced in [8]. It uses generations and random Network Coding to reduce the coding/decoding complexity. This decentralized scheme does not need to know the network topology. However, the packets do not have the same priority and their loss is acceptable consequently there is no need for acknowledgements (ACKs).

An opportunistic coding approach inspired by COPE was proposed in [9] to minimize the delay. The packets to encode are chosen according to their delay threshold in order to minimize the number of packets which miss their deadline.

A Network Coding approach with Multi-Generation Mixing for video communication is presented in [10]. It consists of combining a packet with packets of the same generation and packets of previous generations. It increases the decoding rate for networks with sparse connectivity and high loss rates.

[11] investigates the throughput performance when applying GBNC to scalable multicast. It shows by simulation that the throughput is significantly dependent on the choice of the coding parameters such as generation size, redundancy (or stretch) factor, field size and topology. However, the QoS in terms of delay is not managed and the generation size for each scenario is fixed.

MORE [12] applies Network Coding on an opportunistic routing algorithm named Extremely Opportunistic Routing (ExOR [13]). For a given unicast flow, ExOR finds a multi-path based on the delivery probabilities of the network links. However, a scheduling is needed to prioritize a path on another one. MORE uses random GBNC to cope with this issue. It considers a fixed generation size and does not deal with delay constraints.

For the best of our knowledge, there is only the approach of [14] that adjusts the generation size in a Network Coding context. For a unicast flow, the destination estimates the Round Trip Time (RTT) using an empirical approach and sends it in the ACK packet. The source uses this information to compute the generation size in order to respect the probability that the block-decoding delay is below a given threshold. It is not straightforward to use this approach for multicast flows since the bottleneck is not relevant to one destination. Moreover, the relationship between the given threshold and the throughput is not obvious.

IV. Generation size, throughput and delay

We have explained that the generation size for the GBNC is a crucial parameter. In this section, we detail its impact on network performance such as delay and throughput. This parameter affects the decoding delay that is the duration between sending the first packet (or combination) of a generation at the source side and decoding the last packet of that generation at the destination side. It also affects the overhead (number of ACKs sent to acknowledge the generations). Adjusting the generation size is useful, this is proved by the relationship among throughput, delay and generation size given below.

Let us take the following example: the source node a wants to send Q packets to the destination node b. We consider that the data and ACK packet sizes are respectively L and L'. We assume that the capacity of link $a \rightarrow b$ is equal to $C(bits/s)$ and greater than or equal to the arrival rate at the source. The generation size is k. The number of generations is equal to Q/k, this value indicates the number of ACKs and thus the overhead. Let D be the overall delay for reliably sending the whole file, which means sending all the data packets and receiving all the ACK packets by the source. It is given by :

$$D = \frac{Amount\ of\ sent\ packets + Amount\ of\ ACKs}{Capacity}$$

$$D = \frac{Q \times L + \frac{Q}{k} \times L'}{C}$$

$$\implies C = \frac{Q \times L}{D} + \frac{Q \times L'}{k \times D} = goodput + overhead$$

Assuming a fixed capacity C, we can infer from these formulas the impact of the generation size k on:

- Throughput: the higher is k (the lower is the overhead[2]), the higher is the goodput[3].
- Overall Delay: the higher is k, the lower is the overall delay.

A large generation size appears to be more appropriate to increase the network performance. However, the destination can exploit the native packets of a generation only after receiving and decoding the k coded packets sent by the source. The higher is k, the longer is the decoding delay, the more the information exploited by the destination's higher layers is lately received. That is why this delay is crucial for real time applications.

Due to the variations of the network characteristics (congestion, end-to-end packet delay, network size), a fixed generation size does not guarantee a fixed decoding delay and hence can affect the QoS in terms of delay. Our goal is to build a network-aware approach that adjusts the generation size dynamically in order to maximize the throughput while respecting a fixed decoding delay. The next section explains our approach in details.

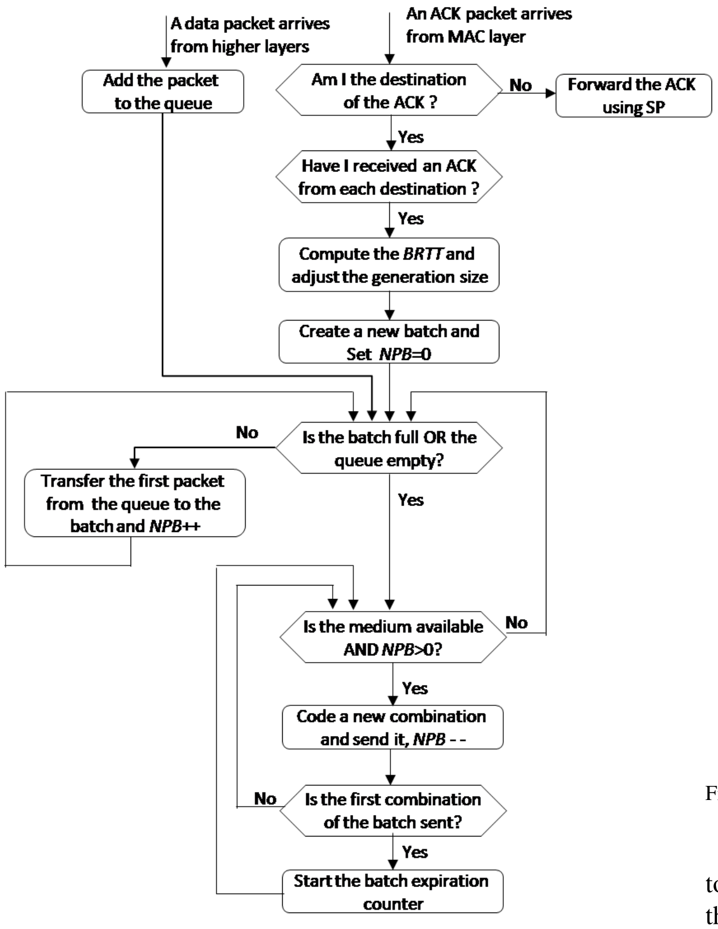
V. Our approach

Our approach performs a feedback control mechanism to adjust the next generation size of a given multicast flow based on some parameters deduced from the last sent generation. We describe below the DYGES method. We present in Figures 1 and 2 detailed diagrams of our MORE-like system for the source and the forwarders/destinations sides.

Before sending the first packet of a multicast flow, the generation size is set to its default value and a new batch is created. A *batch* or *batch matrix* is a table associated to a given multicast flow and that is used to save a copy of the current generation packets.

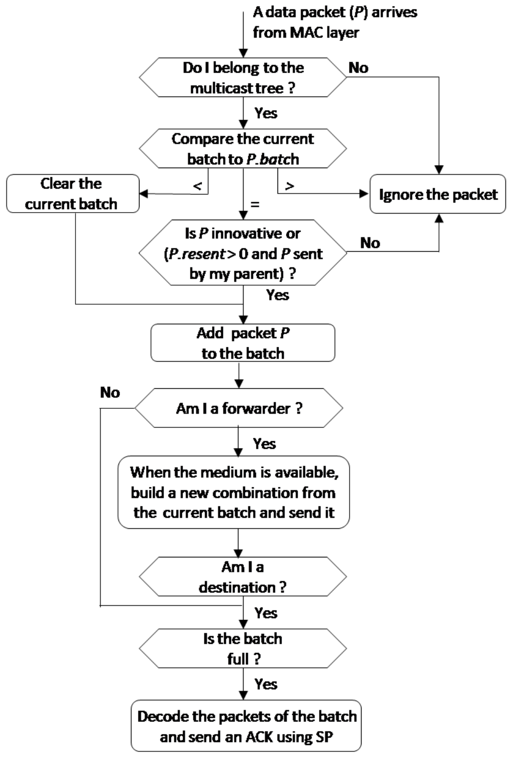
The source (see Figure 1) receives a packet from higher layers and adds it to a temporary queue. This queue is useful when the application layer has a high throughput and thus could overflow the batch matrix. When a new batch is created, packets of the queue are transferred to the batch. However, the number of these packets should not exceed the generation size. Moreover, each time a packet is inserted into the batch, a variable NPB (Number of Packets in the Batch) is increased.

[2] Overhead: troughput of ACK packets.
[3] Goodput: troughput of data packets.



A data packet arrives from higher layers
An ACK packet arrives from MAC layer
Add the packet to the queue
Am I the destination of the ACK ?
No
Forward the ACK using SP
Yes
Have I received an ACK from each destination ?
Yes
Compute the BRTT and adjust the generation size
Create a new batch and Set NPB=0
No
Is the batch full OR the queue empty?
Transfer the first packet from the queue to the batch and NPB++
Yes
Is the medium available AND NPB>0?
No
Yes
Code a new combination and send it, NPB - -
No
Is the first combination of the batch sent?
Yes
Start the batch expiration counter

Fig. 1. Diagram of Sending a packet and Receiving an ACK by the source.



A data packet (P) arrives from MAC layer
Do I belong to the multicast tree ?
No
Yes
Compare the current batch to P.batch
Clear the current batch
<
>
=
Ignore the packet
Is P innovative or (P.resent> 0 and P sent by my parent) ?
No
Yes
Add packet P to the batch
No
Am I a forwarder ?
Yes
When the medium is available, build a new combination from the current batch and send it
Am I a destination ?
Yes
Is the batch full ?
Yes
Decode the packets of the batch and send an ACK using SP

Fig. 2. Diagram of receiving a packet by the forwarders and the destinations.

If the medium is available, as long as there are combined packets to send from the current batch ($NPB > 0$), the source builds a combination using all the packets belonging to the current batch matrix and sends it. We recall that the coefficients are randomly taken from a Galois field (finite field). The value of NPB is then decreased. If the medium is not available, the batch not full and another packet arrives from higher layers, this packet is inserted into the batch and NPB increased. Consequently, this packet will be taken into account to build the combinations to be sent. Note that for the first sent combination of each generation, the source sets a timer for the batch expiration. In case of loss, the timer expiration triggers the retransmission.

Upon receiving a packet P, each intermediate or destination node (see Figure 2) belonging to the multicast tree compares its current batch identifier to the one of P denoted $P.batch$. If it is smaller, this means that the source has already received the ACK for the previous generation and started the new generation with packet P. In this case, the node flushes its batch and resets it with P. Otherwise, if the node has a current batch equal to the one of P it adds it to its batch verifying that the combination P is innovative i.e. linearly independent of the previous combinations it has already received and belonging to that batch. If so and the node is a forwarder (not a leaf in the tree), it is allowed to send a combination. Therefore, when the medium is available, it builds a random combination using the combinations of its batch and sends it. Moreover, if the node is a destination, it checks if it has received the whole batch and if so, sends an ACK asking for a new batch. We want to consider all the packets of a given batch for building a combination in order to maximize the robustness against losses while respecting a steady decoding delay. Consequently, the decoding is launched after receiving the whole generation. Note that the ACK is routed using the Shortest Path (SP) and is prior than data packets when forwarding.

In addition to manage multicast traffic and multiple ACKs, our system differs from MORE in its post batch expiration behavior. Indeed, in the case of retransmission (indicated in the packet with $P.resent > 0$), an intermediate node is allowed to forward a non-innovative combination sent (only[4]) by its parent in the tree. For the targeted applications, we consider that even if a batch expires, it is necessary to resend the batch to make the receiver decoding the generation and retrieving some packets that did not exceed their individual deadline.

When the source (see Figure 1) receives the ACK, it checks whether it has received one ACK from each destination, if so, it resets the batch. To build the new batch, the packets of the queue are prioritized i.e. they are first transferred to the batch before taking into account the arriving packets from the higher layers. Then, the previous sending process is applied.

[4] This solution avoids the flooding phenomenon.

Our system uses the DYGES approach to estimate the next generation size. It computes the $BRTT$ (Batch Round Trip Time), as shown in Figure 3, it is the span between sending the first packet of a generation and receiving the last ACK (from the last destination) for this generation.

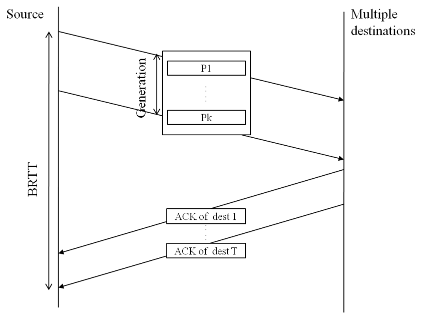


Fig. 3. A source node computing the $BRTT$ after sending a k-sized generation to T destinations.

After computing $BRTT$, DYGES compares it to a threshold $thresh$: if $BRTT$ is below $thresh$, the generation size is increased, otherwise, the generation size is decreased.

$thresh$ depends on the fixed maximum decoding delay required by the application. This feature brings more content awareness to our system.

DYGES is inspired by TCP and aims at adapting dynamically to losses and congestion that make the delay varying in time. It is network-aware i.e. it intends to have a smooth decoding delay independently of the network variations. The next section shows the experimental results.

VI. Simulation results

We implement in our event-based network simulator, written in C++ under Opnet 15.0, a routing layer endowed with Network Coding and an ideal MAC layer. We use the multicast tree algorithm given in [15] to find the nodes implied in the flow forwarding.

We deploy a connected network of $N = 30$ nodes on a square of $1000x1000m^2$. At the beginning of the simulation, a multicast flow is generated and has a lifetime until the end of the simulation. This flow will be used for the study, it has 12 destinations and its tree contains 15 nodes.

Many nodes are chosen to generate unicast flows with randomly chosen destinations. They have a packet inter-arrival according to a Poisson distribution with a parameter $\lambda 1$. Their lifetime follows a Poisson distribution with a parameter $\lambda = 5s^{-1}$ and the flow interarrival follows a Poisson distribution with $\lambda = 20s^{-1}$. The goal of these flows is to vary the network state in order to investigate the behavior of the studied multicast flow.

Each link is lossless and transmits a packet in $1ms$. The sending buffer of an intermediate node (at the MAC layer) has an unlimited capacity. To build a combination the coefficients are chosen in a Galois Field GF[251]. The batch expiration time is set to $200ms$ and $thresh = 95ms$. Here, the increment used for adjusting the generation length is ± 1.

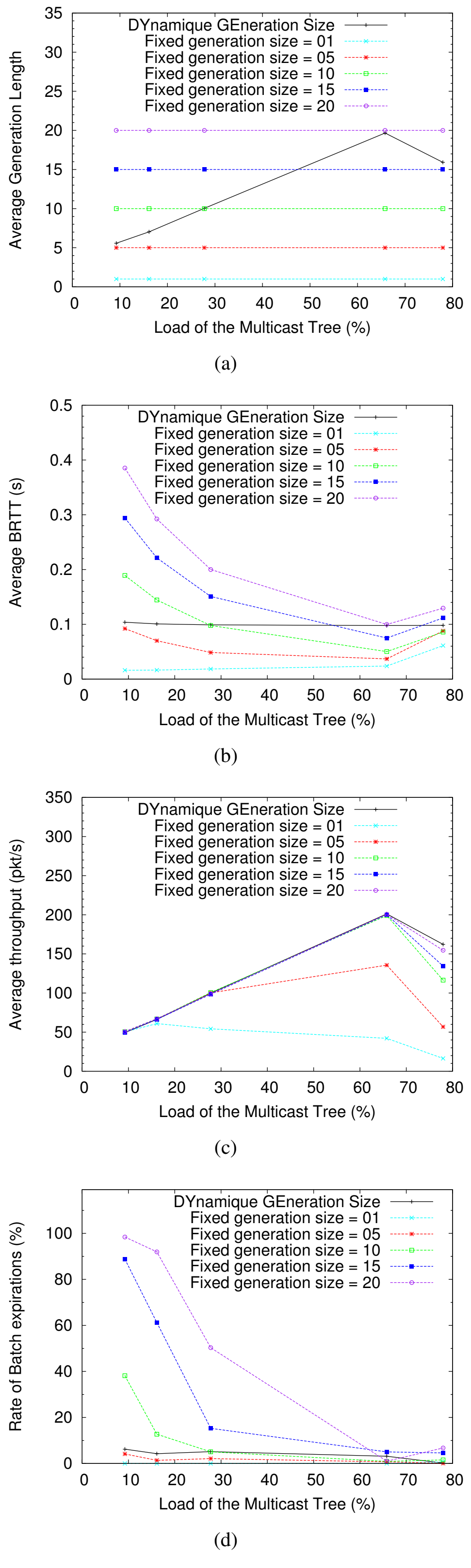


Fig. 4. Evolution of (a) the average generation size, (b) average $BRTT$, (c) average throughput and (d) rate of batch expirations according to the load of the multicast tree.

We focus on the load of the studied multicast tree, we vary the network load by varying $\lambda 1$ and the number of flows. We consider the reference (a load of 100%) when all the nodes of the tree are sending packets. The simulation duration is $100s$.

Figure 4 shows the simulation results for the studied flow. We compare DYGES with different fixed generation size systems to investigate the impact on delay, throughput and batch expiration rate.

- For the low load ($\leq 28\%$), the packet inter-arrival is much lower than the network capacity and this makes the receivers waiting idly for the complete generation. This explains the high $BRTT$ obtained using a high generation size on Figure 4(b) and the number of their $BRTT$ expirations (see Figure 4(d)). The throughput of the different approaches, plotted on Figure 4(c) is identical except for the generation size that is equal to one, it has a very high overhead because each packet is ACKed individually. Even if it appears that the generation size of 10 has a lower delay than DYGES while having the same throughput, the goal of DYGES is not to find the optimal values of delay and throughput, but to keep the delay as steady as possible without dramatically losing performance in terms of throughput.
- When the network load increases ($28\% < load \leq 65\%$), the idle time decreases leading to still decrease the $BRTT$ of the fixed generations. The small generation size has a low delay but at the cost of throughput because it does not maximize the benefits of Network Coding. As for DYGES, it keeps the delay steady and the throughput at its maximum value. To do so, it adapts the average generation size until 19.7 (see Figure 4(a).).
- For the high load ($> 65\%$), there is more congestion engendered by the other flows and thus a longer end-to-end packet delay. Consequently, the $BRTT$ delay increases and the throughput decreases. A large generation size leads to more $BRTT$ expirations (compared to the load of 65%). As for DYGES, it decreases its average generation size in order to, on the one hand, have a high throughput and on the other hand, still keep the delay around the threshold.

We can conclude that DYGES adapts dynamically the generation size to keep the decoding delay steady and thus to guarantee the QoS in terms of delay under the different loads, this shows its network-awareness.

VII. Conclusion

In this paper we have studied the issue of QoS in terms of delay for GBNC systems. Most of these systems consider a fixed generation size. However, the delay can vary significantly according the network state and then degrade the application performance. Therefore, we proposed DYGES, a network-aware approach that adjusts the generation size according to the network variations (network size, congestion, loss). The simulation results show that our approach adapts the generation size in order to maintain the decoding delay steady while keeping a high throughput.

We have considered a fixed generation size increment, it would be interesting to analyze DYGES behavior when using a dynamic increment. However, this increment should be carefully computed to avoid frequent $BRTT$ oscillations that may increase the batch expirations and thus decrease the network performance. Moreover, we aim at investigating more parameters such as the field size, redundancy factor and default generation size. Since DYGES is *intra-flow* which means only packets belonging to the same generation of one flow are mixed together, it could be interesting to study the behavior of an *inter-flow* approach mixing packets of different flows.

VIII. Acknowledgments

This work was supported in part by Orange Labs, the R&D department of France Telecom. We would like to thank Dr. Hassnaa Moustafa at Orange Labs for her guidance and precious discussions.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.
[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, 2003.
[4] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *IEEE International Symposium on Information Theory (ISIT'03)*, 2003, p. 442.
[5] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'06)*, New York, NY, USA, 2006, pp. 243–254.
[6] S. Omiwade, R. Zheng, and C. Hua, "Practical localized network coding in wireless mesh networks," in *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08)*, June 2008, pp. 332–340.
[7] J. Le, J. C. S. Lui, and D.-M. Chiu, "Dcar: Distributed coding-aware routing in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 4, pp. 596–608, 2010.
[8] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *41st Annual Allerton Conference Communication, Control, and Computing*, 2003.
[9] Z. Dong, C. Zhan, and Y. Xu, "Delay aware broadcast scheduling in wireless networks using network coding," in *Second International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC'10)*, 2010, pp. 214–217.
[10] M. D. Halloush and H. Radha, "Network coding with multi-generation mixing: Analysis and applications for video communication," in *42nd Annual Conference on Information Sciences and Systems (CISS'08)*, 2008, pp. 515–520.
[11] J.-P. Thibault, S. Yousefi, and W.-Y. Chan, "Throughput performance of generation-based network coding," in *10th Canadian Workshop on Information Theory (CWIT'07)*, 2007, pp. 89–92.
[12] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'07)*, 2007, pp. 169–180.
[13] S. Biswas and R. Morris, "Exor: Opportunistic multi-hop routing for wireless networks," in *ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'05)*, 2005, pp. 69–74.
[14] K. Xu, F. Zhang, C. Rong, B. Dai, and B. Huang, "Block size estimation for time-sensitive applications under wireless network coding," in *Second International Conference on Future Generation Communication and Networking (FGCN'08)*, 2008, pp. 321–324.
[15] A. Penttinen, "Efficient multicast tree algorithm for ad hoc networks," in *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'04)*, 2004, pp. 519–521.