Forty-Fifth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
September 26-28, 2007

Real Network Codes

Sachin Katti skatti@mit.edu
Saurabh Shintre shintre@ee.iitb.ac.in
Sidharth Jaggi sidharth@jaggi.name
Dina Katabi dk@mit.edu
Muriel Medard medard@mit.edu

Abstract– Network coding has been shown to improve throughput and reliability in a variety of theoretical and practical settings. But it has had limited success in areas like sensor networks due to it's two limitations. First, network codes are ”all-or-nothing” codes; the sink cannot decode any information unless it receives as many coded packets as the original number of packets. Second, sensor networks often measure physical signals which show a high degree of spatial correlation; present network coding techniques cannot perform in-network lossy compression to take advantage of the spatial correlation.

This paper presents ”Real” Network Codes that are linear over real fields. We build on recent results from Compressed Sensing to develop new codes which can be decoded to get progressively more accurate approximations as more coded packets are received at the sink. Further, they can compress distributed correlated data inside the network without requiring that the nodes know how the data is correlated. Thus, Real Network Codes combine two exciting but hitherto separate areas, Network Coding and Compressed Sensing, allowing them to keep the advantages of network coding, but also make them capable of finding low distortion approximations with partial information and perform distributed compression of correlated data.

1 INTRODUCTION

Network Coding breaks with the traditional assumption and allows nodes to mix packets inside the network. Theoretically this has been shown to achieve the multicast capacity of the network. Recent work [17, 6] has shown that significant gains are also realized in practice. Specifically in wireless networks large throughput gains are obtained due to network coding's ability to exploit wireless broadcast and take advantage of opportunistic receptions. Perhaps the most important reason for its success is its simplicity, simple random linear codes inside the network suffice to achieve the above mentioned gains.

But network coding still leaves a lot to be desired. Network Coding combines packets over finite fields inside the network which gives it a ”All or Nothing” property. If n packets are combined using network coding, the sink has to receive atleast n packets in order to be able to recover the n original packets. Thus if the sink receives $n-1$, it cannot recover any information. Real world applications like sensor networks, video, audio etc cannot tolerate such a hard constraint on their performance; they want a more graceful degradation. For example, a sensor network operator would not like to be left with no information if for some reason the network is not able to deliver n packets; ideally he would like to recover as much partial information as possible. Network Coding as constructed right now cannot deliver such graceful degradation in its performance, thus limiting its applicability.

This paper introduces Real Network Codes (RNC), a new code design which keeps the good properties of traditional network codes while getting rid of the undesirable properties. RNCs are designed to deliver accurate approximations to the original data when less than n packets have been received. Further RNC can perform in network compression on correlated distributed data. But it retains the good properties of traditional network coding, i.e., it can efficiently take advantage of wireless broadcast and maximize multicast throughput.

The main insight behind RNC is to recognize that practical signals are described best in the real domain, so one should perform the network coding in the real field itself. Thus nodes treat packet contents as finite precision real numbers instead of elements of a finite field. They then perform random linear coding over these finite precision real numbers. The destination gets a set of real numbers which are the result of a real random linear transform applied to the original information. RNC then uses powerful decoding algorithms from Compressed Sensing [5, 4, 9] to recover accurate approximations to the original data even when it has received far fewer than n linear combinations. RNC behaves like standard network codes when n packets are received, it can decode the original information completely. But RNC can also construct approximations with low distortion when fewer than n packets have been received; achieving the desired graceful degradation in its performance.

This paper makes the following contributions

- A new network coding technique, RNC, which delivers progressively more accurate approximations as more packets are received at the sink, thus getting rid of the ”All or Nothing” property of traditional network codes.

- RNC also performs in-network compression of correlated distributed data while maintaining the nice architectural separation of traditional network coding. Interior nodes perform simple random linear coding without even being aware of the correlation that exists in the data, but the sink uses sophisticated decoding algorithms which exploit the

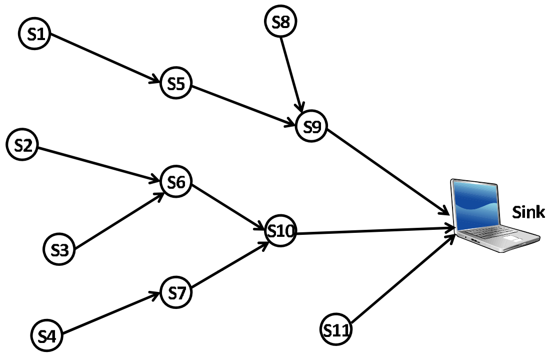

Figure 1—Sensor network measuring temperature. Present approach organizes them in a tree leading to the sink to deliver measurements. Network Coding is more efficient because it can take advantage of opportunistic receptions resulting from wireless broadcast.

correlation to decode accurately using fewer number of packets. Thus the interior nodes end up compressing the data inspite of using simple random linear codes.

- We present a low-complexity and practical technique for minimizing the number of transmissions required in a sensor network while using Real Network Codes. The technique adapts to network congestion and delivers an approximation to the sink whose accuracy is commensurate with the available bandwidth in the network.

2 Example

We illustrate RNC through an example from sensor networks. Consider a distributed array of sensors shown in Fig. 1 which are measuring a physical signal like temperature. There is a sink which wishes to collect all the measurements periodically from the sensor network. The sensors are connected to each other via wireless links. The sensors collaborate in relaying each other's measurements back to the sink.

The present approach builds a tree of links leading to the sink from all the sensors in the network. Sensors transmit their measurements along this tree to the sink. Each sensor uses retransmissions to its next hop on the tree to guarantee reliable delivery. This approach has a few nice properties. First, it is simple and easy to build. Second, even if the sink fails to get all the measurements from the sensors, it still gains partial information about the physical signal. Thus every received packet is useful.

But the above approach ignores the broadcast nature of wireless links. When a sensor broadcasts its measurement, several of its neighbor sensors are likely to receive the measurement. But in the current approach only the next hop on the tree is supposed to relay the measurement. Thus if the next hop doesn't receive the broadcast but some other sensor overhears it, the tree approach cannot take advantage of the opportunistic reception and has to retransmit. For example, in Fig. 1, when S2 transmits, if S5 receives it but S6 which is the next hop on the tree doesn't, S2 has to retransmit, wasting S5's opportunistic reception.

Network Coding can be used to address the above shortcoming. With network coding, a node transmits random linear combinations of all the measurements it has received. Opportunistic receptions are therefore not wasted; sensors re-encode overheard measurements when they transmit their packets. In Fig. 1, S5 can take advantage of the opportunistic reception and S2 does not have to retransmit. Network Coding guarantees that the sink can recover the n sensor measurements once it gets any n random linear combinations of the measurements. Both theoretical and practical work has shown that network coding can lead to significant reduction in the number of transmissions [6, 21]. But network coding imposes a significant burden on the sensor network. It has to ensure that the sink receives atleast n linear combinations, anything less and the sink will not be able to recover anything. This is the well known "All or Nothing" property of network coding. But sensor network designers want a graceful degradation in the performance, the sink should be able to get an estimate of the physical signal even when it does not receive n packets.

Further, both approaches above share another shortcoming. Sensor networks sense physical signals which show continuous behavior across space and time, thus colocated sensors are likely to see correlated measurements. Both approaches ignore this correlation and treat each measurement as an independent sample. Significant savings could be had if sensors could exploit this spatial correlation by compressing the measurements as they flow through the network.

Real Network Codes (RNC) keep the good properties of both approaches while addressing their shortcomings. The main insight behind RNC is to recognize that source data is naturally embedded in the real field, thus to get a good distortion behavior one should also code data in the same real field. Traditional network coding encodes real data over finite fields leading to the all or nothing property. But when data is coded over the real field, the sink can find approximations to the original physical signal even when it has received less than n packets. The approximation algorithm is based on the intuition that the colocated sensor measurements can be treated as colocated pixels in an image since they display the same spatial correlation behavior. Due to the spatial correlation, the wavelet transform of an image is sparse, thus using only the largest wavelet coefficients to represent the image gives an efficient and accurate approximation to the original image. Similarly, we can use the largest wavelet coefficients to approximate the sensor measurements.

RNC leverages algorithms from the recent field of Compressed Sensing [5, 4, 9] to find the largest wavelet coefficients from random linear combinations of the original data. The decoding algorithms recover the wavelet coefficients in descending order of magnitude as more and more packets arrive at the sink. Thus it achieves the same graceful degradation of performance as a wavelet representation would have. Second, since wavelets incorporate the spatial correlation to compress data, RNC's ability to recover the largest coefficients quickly from far fewer than n linear combinations implies that it automatically performs in-network compression of the data without even knowing the correlation that exists among the sensors. RNC achieves the above properties while keeping a clean architectural separation; low complexity sensors do very simple encoding tasks on whatever packets they receive and do not need to co-ordinate on how to deal with correlated data; while the more powerful sink can employ smarter decoding algorithms to take advantage of the spatial correlation and deliver accurate approximations when fewer than n packets have been received.

3 RNC Architecture

RNC is applicable in a large variety of settings, but for ease of exposition we describe it is terms of a sensor network. We focus on the case where there is a single sink wishing to estimate the distributed physical signal being measured by the sensor network, but the techniques can be easily extended to the multicast scenario of multiple sinks. The number of sensors is n.

3.1 Sensors/Sources

The sensors take measurements periodically in batches which are indexed by t. We call the original measurement at a sensor a *native measurement*. The native measurement at sensor i in batch t is represented by x_{it} and the distributed native measurement vector is $\vec{x_t}$. The sink wants to collect these native measurements for this batch t.

3.2 Relaying

Sensors listen to all transmissions. When a sensor hears a transmission, it checks whether it contains new information. Technically speaking, the received packet contains useful information if the coded measurement is linearly independent from the coded measurements the nodes has previously received from the batch. If the received coded measurement contains new information, the sensor keeps it in its buffer.

When the wireless MAC signals an opportunity to send, the sensor broadcasts a new coded measurement. A coded measurement is defined as $y_{jt} = \sum_i c_i x_{it}$ where c_i is a random real number, i.e., it is a random linear combination of native measurements. The sensor creates a new coded measurement by randomly combining its native measurement with all the overheard coded measurements. The random coefficients are picked from a Bernoulli $1/-1$ distribution. The vector $\vec{c_j} = (c_1, \ldots, c_n)$ is called the code vector and is transmitted in the header of the coded measurement packet being transmitted.

3.3 Sink

For each coded measurement is receives, the sink checks whether it contains useful information, i.e., it is linearly independent from previously received measurements. The sink keeps the useful coded measurements and proceeds to decode. The vector of coded measurements received at the sink can be written efficiently as

$$\vec{y_t} = A\vec{x_t} \tag{1}$$

where A represents the random linear transform the network has applied over the native measurements $\vec{x_t}$.

If the sink receives n linearly independent coded measurements, i.e., if A is invertible, its job is simple. It decodes $\vec{x_t} = A^{-1}\vec{y_t}$. This is exactly the same as standard network coding.

What if the sink receives less than n coded measurements? The primary goal of RNC is to be able to decode partial information about $\vec{x_t}$ when less than n coded measurements have been received. It obviously cannot reconstruct the original measurements exactly, but instead it finds an approximation which is close to the best lossy representation of the original signal. We first describe what the best lossy description of a signal is and then how RNC finds it.

3.3.1 *Lossy Representation*

How does one efficiently describe signals in a lossy fashion? The answer is to transform the signal into a basis in which it is sparse and then use the largest coefficients to represent the signal. For example, suppose you have a multitone audio signal which consists of a few significant frequency components along with noise. Its Fourier transform will be sparse, there will be significant components at the frequencies which make up the signal and the rest of the frequency components will be close to zero. A very efficient but lossy representation of such a signal is to keep the significant frequency components and throw away the rest. The same principle is used to represent most practical signals since they can usually be represented in a sparse fashion in some basis. For audio it is the Fourier basis, for images it is the wavelet basis and so on.

Formally, we can fix an orthonormal transform $\phi \in R^{n\times n}$ consisting of a set of orthonormal basis vectors $\{\phi_1, \ldots, \phi_n\}$. ϕ can be, for example, a wavelet or a Fourier transform. The transform coefficients $\theta = [\phi_1^T\vec{x_t}, \ldots, \phi_n^T\vec{x_t}]$ of the native measurement vector can be ordered in magnitude, so that $|\theta_{(1)}| \geq |\theta_{(2)}| \geq \ldots \geq |\theta_{(n)}|$. The best k-term approximation keeps the k-largest coefficients and discards the remaining as zero. The approximation error is $||\vec{x_t} - \vec{x_t'}||_2^2 = ||\vec{\theta} - \vec{\theta'}||_2^2 = \sum_{i=k+1}^{n} |\theta|_{(i)}^2$. The magnitude of the transform coefficients of real-world signals typically exhibit a power law decay. That is, the i^{th} largest transform coefficient satisfies $|\theta|_{(i)} \leq Ri^{-1/p}$ for each $1 \leq i \leq n$, where R is a constant and $0 < p < 1$. The approximation error for such signals can be shown to be

$$||\vec{x_t} - \vec{x_t'}||_2^2 = ||\vec{\theta} - \vec{\theta'}||_2^2 = \alpha_p R k^{-1/(p+1/2)} \quad (2)$$

To represent the distributed native sensor measurement vector $\vec{x_t}$ efficiently, one has to first find the basis ϕ in which it is sparse. Image compression provides us the intuition here. If we think of the sensor network as two-dimensional array, the readings at each sensor can be thought of as pixels of an image. They have the same properties as typical images, readings are correlated among geographically close sensors. Images are efficiently compressed by transforming them using the wavelet transform and keeping the significant wavelet coefficients. Similarly an appropriately designed wavelet transform can be used to find a sparse representation of the sensor measurement vector $\vec{x_t}$.

3.3.2 *Recovering a lossy representation*

How does one recover the significant wavelet coefficients of the distributed measurement vector $\vec{x_t}$ from random linear combinations of the original measurements which RNC provides? RNC takes advantage of powerful techniques from Compressed Sensing [5, 4, 9] to find the significant wavelet coefficients. Compressed Sensing provides algorithms which can recover a very good approximation to the original signal if it is sparse in some basis. Specifically, $O(k \log n)$ random linear combinations of the native measurement vector can produce an approximation with error comparable to the best k-term approximation error using the k largest transform coefficients. In our case, the transform is the wavelet transform and the approximation error will be that of the wavelet representation.

More concretely, consider the following random linear transform $\psi \in R^{k \times n}$ containing i.i.d entries

$$\psi_{ij} = \begin{cases} +1 & \text{w. p.} \quad 1/2 \\ -1 & \text{w. p.} \quad 1/2 \end{cases} \quad (3)$$

Then the k random linear combinations $\frac{1}{\sqrt{n}}\psi\vec{x_t} \in R^k$ produce an approximation $\vec{x_t'}$ of the native measurement vector $\vec{x_t}$ with distortion

$$||\vec{x_t} - \vec{x_t'}||_2^2 = \beta_p R (k/\log n)^{-1/(p+1/2)} \quad (4)$$

assuming the power law decay of the wavelet transform coefficients as described above and β_p is some function of p.

The approximation can be found by using the following linear program

$$\arg\min ||\vec{\theta}||_1 \quad (5)$$

subject to the constraints

$$\begin{aligned} \vec{y} &= \psi\vec{x_t} \\ \vec{\theta} &= \phi\vec{x_t} \end{aligned} \quad (6)$$

The above problem can be solved using standard linear programming techniques which has $O(n^3)$ complexity. Faster algorithms based on Orthogonal Matching Pursuit [8] are available to solve the above problem, but they require slightly higher number of coded measurements.

The above guarantee is for the specific random linear transform described by ψ. The random linear transform A applied by the network in RNC in general is arbitrary and depends on the network structure. In practice, the transform A behaves similarly for typical small networks and satisfies the above guarantees. For larger networks there is information loss due to finite precision arithmetic, which we discuss further in §4.2.

3.3.3 *Accurate Approximations and Multicast*

RNC ensures that if the sink receives $O(k \log n)$ coded measurements, it can find the top k wavelet coefficients w.h.p. Thus as the sink receives more measurements, it recovers more wavelet coefficients, in descending order of their magnitude. Since practical signals exhibit a power law behavior in their sparse representation, the top wavelet coefficients carry most of the energy of the original signal. As we recover more and more of the top wavelet coefficients the distortion of our estimate drops similar to a power law distribution PDF.

Second, RNC decouples the performance of multiple sinks when the data is being multicast. Traditional network coding makes all sinks equal to the worst sink, all sinks get data at the same rate as the one with the lowest capacity. While RNC cannot increase the rate to sinks with higher capacity; it ensures that they get better approximations to the original signal. This is because they get coded measurements faster than the lower capacity sinks, thus the distortion of their reconstructed signal drops faster.

3.3.4 *Spatial Correlation*

RNC automatically does in-network compression of the correlated sensor measurements. It stems from the fact that the wavelet representation captures the spatial correlation that exists in the native measurement vector. Due to the correlation, the wavelet transform vector is sparse, and RNC needs only a few coefficients to recover the original signal with high accuracy. For example, if the native measurement vector $\vec{x_t}$ can be represented with 99% accuracy using the top $l << n$ transform coefficients, RNC needs only $O(l \log n) << n$ coded measurements to recover an approximation with the same accuracy. The sensors inside are oblivious to the spatial correlation, RNC takes advantage of the spatial correlation directly at the sink.

4 Practical Challenges

In §3 we have described the general design of RNC. But for them to be practical, we have to address 2 additional challenges which we discuss below.

4.1 Who should transmit and how many?

In the current tree based approach in sensor networks, a node keeps transmitting a packet until the nexthop receives it, or the number of transmissions exceeds a particular threshold, at which time the node gives up. However with RNC we want to exploit opportunistic receptions, all nodes closer to the sink than the current transmitter are potential next hops and may participate in forwarding the packet. How many transmissions are necessary to ensure that at least one node closer to the sink has received the packet?

We provide a heuristic-based practical solution to the above problem. Our solution has the following desirable characteristics: 1) It has low complexity. 2) It is distributed. 3) It is practical, i.e., it makes no assumptions about the channel conditions in the sensor network and only relies on average loss measurements.

(a) Approach: Since bandwidth and energy are scarce in a sensor network, we wish to minimize the number of transmissions required inside the network. Let the distance from a sensor i to the sink d be the number of transmissions necessary to deliver a packet from that sensor to the sink, i.e. the node's ETX [7]. We propose the following heuristic to route a packet to the sink, when a sensor transmits a packet the sensor closest to the sink in ETX distance among all the sensors that received it should forward that packet. This heuristic reduces the expected number of transmissions needed to deliver the packet, and thus reduces the overall number of transmissions.

Formally, let N be the number of sensors in the network. For any two sensors, i and j, let $i < j$ denote that sensor i is closer to the sink than sensor j, or said differently, i has a smaller ETX than j. Let p_{ij} denote the loss probability in sending a packet from i to j. Let z_{is} be the expected number of transmissions that forwarder i must make to route one packet from the source, s, to the sink, d, when all sensors follow the above routing heuristic. In the following, we assume that wireless receptions at different sensors are independent, an assumption that is supported by prior measurements [24, 22].

We focus on delivering one packet from a sensor to sink. Let us calculate the number of packets that a forwarder j must forward to deliver one packet from sensor, s to sink, d. The expected number of packets that j receives from sensors with higher ETX is $\sum_{i>j} z_{is}(1-p_{ij})$. For each packet j receives, j should forward it only if no sensor with lower ETX metric hears the packet. This happens with probability $\prod_{k<j} p_{ik}$. Thus, in expectation, the number of packets that j must forward, denoted by L_j, is:

$$L_{js} = \sum_{i>j} (z_{is}(1-p_{ij}) \prod_{k<j} p_{ik}). \tag{7}$$

Note that $L_s = 1$ because the sensor generates the packet.

Now, consider the expected number of transmissions a sensor j must make. j should transmit each packet until a sensor with lower ETX has received it. Thus, the number of transmissions that j makes for each packet it forwards is a geometric random variable with success probability $(1 - \prod_{k<j} p_{jk})$. This is the probability that some sensor with ETX lower than j receives the packet. Knowing the number of packets that j has to forward from Eq. (7), the expected number of transmissions that j must make is:

$$z_{js} = \frac{L_{js}}{(1 - \prod_{k<j} p_{jk})}. \tag{8}$$

The above calculation was for a single sensor being considered as the source. But since we want to estimate the entire sensor field, we have to ensure that adequate number of measurements from all sensors are available at the sink. Thus the above process is repeated for each sensor as a source and the total number of transmissions that a sensor i must make are added up for all sources, i.e. $z_i = \sum_s z_{is}$. The calculations depend only on the loss probabilities of the links in the sensor network. Most link state routing protocols already disseminate these measurements to all the sensors, thus sensors can compute the number of transmissions easily.

Calculating the number of transmissions a sensor needs to make is not sufficient, the sensor also needs to know when to transmit. Further the above calculation is for making sure the sink receives n coded measurements, but in practice due to the spatial correlation far fewer measurements may suffice. We need a way to *clock* transmissions at each sensor. We do this via two mechanisms; first the sink sends an Ack when it determines it has a good reconstruction of the original signal. Second, we clock the transmissions at each sensor depending on the number of coded measurements it receives. Specifically, we assign each node *TXCredit* for every packet it receives, this reflects the number of transmissions a node should make for every reception. For each packet sent from sensor s to sink, sensor i receives $\sum_{j>i} p_{ji} z_{js}$, where z_{js} is the number of transmissions made by sensor j for this sensor source s and p_{ji} is the delivery probability from j to i. Thus, the TX_credit of node i is:

$$\text{TX\_credit}_i = \sum_s \frac{z_{is}}{\sum_{j>i} z_{js} p_{ji}}. \tag{9}$$

Each sensor therefore keeps a `credit counter`. When it receives a coded measurement from a node upstream it updates its credit counter by its *TXCredit*. When it has enough credit to make one transmission and the wireless MAC signals an opportunity to send, it sends out a new coded measurement. This ensures that sensors do not transmit useless coded measurements. Second the Ack from the sink ensures that nodes do not transmit more than necessary.

4.2 What about finite precision?

RNC does coding over the real field inside the sensor network. But since we can represent reals only with finite precision, rounding off errors may accumulate as the coded measurements propagate through the network. We analyze this in detail in an accompanying paper [25] and show the amount of information loss that can occur due to finite precision arithmetic when we are trying to decode the complete original information. We show that with random coding the information loss is roughly logarithmic in network size. We also provide insights on designing coding techniques that minimize the loss. But since the native measurements are correlated, the sink needs far fewer coded measurements to get an accurate representation, further reducing the number of finite precision operations that occur inside the network. Therefore in practice the information loss is going to be even smaller.

5 Related Work

Prior work can be divided into two categories: network coding and compressed sensing. We review each of them below.

5.1 Network Coding

Work on network coding has started with a pioneering paper by Ahlswede et al. that establishes the value of coding in the routers and provides theoretical bounds on the capacity of such networks [1]. The combination of [19, 18, 13] shows that, for multicast traffic, linear codes achieve the maximum capacity bounds, and coding and decoding can be done in polynomial time. Additionally, Ho et al. show that the above is true even when the routers pick random coefficients [11]. Researchers have extended the above results to a variety of areas including content distribution [10], secrecy [3, 12], and distributed storage [14].

Of particular relevance is prior work on wireless network coding [20, 16, 17, 6]. This work can be divided into three classes. The first is theoretical; it extends known information theory bounds from wired to wireless networks [20, 12]. The second is simulation-based; it designs and evaluates network coding protocols using simulations [23, 26]. The third is implementation-based; it uses implementation and testbed experiments to demonstrate achievable throughput gains for sensors and mesh networks [17, 15, 6]. This paper especially uses insights from [6] to determine how and when sensors should transmit, but differs from it and other prior work in two main ways. First, RNC provides a new network coding technique which can deliver accurate approximations with partial information, while prior work has an "All or Nothing" behavior. Second, it performs in-network compression of correlated data without any knowledge of the correlation inside the network, while prior work either does not perform any compression or assumes that the extent of correlation is known at the sources.

5.2 Compressed Sensing

The idea of obtaining efficient signal representations via random projections has very recently received a great deal of attention in the signal processing community, beginning with the ground breaking papers [5, 4, 9]. There has been work on using compressed sensing to compress distributed correlated sources [2] in scenarios like sensor networks. This work differs from prior work in two important ways. This is the first work to make the connection between Compressed Sensing and Network Coding and show how one can exploit the synergy between these two ideas to get the best of both. Here the network co-operatively takes the measurement, whereas in previous work the measurements were taken at the sensors independently and then the network just acted as a conduit for these measurements to the sink. The second main difference is that this allows separation between source compression and in-network compression. Sensors can employ standard techniques like MP3 etc to compress their own data, we can then use RNC on this compressed data to exploit the spatial correlation. This turns out to be an efficient division of labor since lossy transform-based source coding techniques are already very efficient and compressed sensing techniques are worse by a logarithmic factor which is quite expensive in practice. RNC uses compressed sensing techniques only to exploit the spatial correlation between distributed sensors where it is suited, since then the sensors do not need to know the exact correlation structure and can take advantage of network coding.

6 Conclusion and Future Work

Network Coding and Compressed Sensing are powerful ideas which until now have appeared unrelated. Our work combines them in a natural fashion to provide Real Network Codes which remove the "All or Nothing" constraint of traditional network codes and can perform in-network compression of correlated data. RNCs can therefore be used in new applications such as sensor networks, audio streaming etc to which traditional network coding is ill-suited. RNC also opens up many avenues of future research. Interesting questions include designing codes which do not suffer a large amount of information loss due to finite precision arithmetic, computing functions on correlated data, investigating iterative decoding techniques and designing sparse network codes.

References

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network Information Flow. In *IEEE Trans. on Information Theory*, Jul 2000.

[2] D. Baron, M. Wakin, M. Duarte, S. Sarvotham, and R. Baraniuk. Distributed compressed sensing. In *Allerton*, 2005.

[3] N. Cai and R. W. Yeung. Secure Network Coding. In *ISIT*, 2002.

[4] E. Candes and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies?, Apr 2006.

[5] E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.

[6] S. Chachulskit, M. Jennings, S. Katti, and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *SIGCOMM*, 2007.

[7] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A

high-throughput path metric for multi-hop wireless routing. In *MOBICOM*, 2003.

[8] D. Donoho, Y. Tsaig, I. Drori, and J.-C. Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit, Mar 2006.

[9] D. L. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[10] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. In *INFOCOM*, 2005.

[11] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger. On randomized network coding. In *Allerton*, 2003.

[12] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard. Resilient Network Coding In The Presence of Byzantine Adversaries. In *INFOCOM*, 2007.

[13] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Trans. on Information Theory*, 2003.

[14] A. Jiang. Network Coding for Joint Storage and Transmission with Minimum Cost. In *ISIT*, 2006.

[15] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth Codes: Maximizing Sensor Network Data Persistence. In *SIGCOMM*, 2006.

[16] S. Katti, D. Katabi, W. Hu, H. S. Rahul, and M. Médard. The importance of being opportunistic: Practical network coding for wireless environments. In *Allerton*, 2005.

[17] S. Katti, H. Rahul, D. Katabi, W. H. M. Médard, and J. Crowcroft. XORs in the Air: Practical Wireless Network Coding. In *SIGCOMM*, 2006.

[18] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Trans. on Networking*, 2003.

[19] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Trans. on Information Theory*, Feb 2003.

[20] D. S. Lun, M. Médard, and R. Koetter. Efficient operation of wireless packet networks using network coding. In *IWCT*, 2005.

[21] D. S. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee. Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding. In *IEEE INFOCOM*, 2005.

[22] A. K. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *MOBICOM*, 2005.

[23] J. S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Médard. Codecast: A network-coding based ad hoc multicast protocol. *IEEE Wireless Comm. Magazine*, 2006.

[24] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-based models of delivery and interference. In *SIGCOMM*, 2006.

[25] S. Shintre, S. Katti, S. Jaggi, B. K. Dey, D. Katabi, and M. Medard. Real and complex network codes: Promises and challenges. In *NetCod*, 2008.

[26] J. Widmer and J.-Y. L. Boudec. Network Coding for Efficient Communication in Extreme Networks. In *SIGCOMM WDTN*, 2005.