Convolutional Network Codes for Reliable Point-to-Point Wireless Communication

Samantha R. Summerson
Rice University
Houston, Texas 77005
Email: srsummerson@rice.edu

Anuj Batra
Distinguished Member of Technical Staff
Smart Grid R&D Systems
Texas Instruments Incorporated
13532 N. Central Expressway, Dallas, TX 75243
Email: batra@ti.com

Abstract—This paper considers the idea of using convolutional network codes to code across blocks of packets at the transmitter in order to increase the reliability of a wireless link. The encoding and decoding operations exploit traditional convolutional codes and Viterbi decoders, respectively, and are defined in such a way to be compatible with existing technologies. We present results for the packet error rate (PER) of various convolutional network coding schemes and show that there is a steep decline in the PERs as the signal-to-noise ratio (SNR) increases. We also show that it is possible to achieve large coding gains when a soft network decoder is used.

I. INTRODUCTION

Network coding is known to increase the spectral efficiency of wired networks [1]. Nodes in the network forward a function of the received bits, where the function is defined by the network code and is chosen such that the original bits can be uniquely determined at the receiver from the final set of network coded bits received. This reduces the average number of transmissions needed, since a function of the bits is transmitted rather than each bit individually, and subsequently reduces the number of retransmissions if there are errors or erasures.

Additionally, network coding can be used to improve the reliability of a wireless network [2]–[6]. Wireless networks are often plagued by packet failures due to noise and fading in the channel. We propose to exploit network coding techniques to improve the reliability of a point-to-point wireless link by implementing coding at the network layer.

Previous works have considered convolutional network coding as either performing network coding at a given node using a subset of all received packets [7], [8] or as a result of performing typical network coding in a cyclic network [9]. In both cases, the convolutional network code is tied to the network topology. We propose to use a convolutional code across packets as a way to increase the robustness of a wireless link; since point-to-point communications is considered, the convolutional network code is not dictated by a specific network topology, rather it can be designed with only reliability in mind. This type of error correcting codes are used extensively to achieve reliable communication or data transfer. The design of the encoder and the effect of the convolutional code parameters, such as constraint length, will be discussed in the following sections. By encoding over packets, convolutional network coding can reduce the PER, especially when a soft network decoder is used at the receiver.

II. SYSTEM AND CHANNEL MODEL

We propose to implement network codes to increase the reliability of an existing point-to-point wireless link, with the network code operating at the network layer. The PHY and MAC layers of the wireless system remain unchanged, except for the cyclic redundancy check (CRC) block. The transmitter and receiver are depicted in Figure 1.

In Figure 1, \( K \) message packets are encoded at the network layer to produce \( N \) network coded packets through an encoding process discussed in Section III. The network coded packets are then passed down through the MAC and PHY for wireless transmission. Two types of decoding at the receiver will be considered: hard decoding and soft decoding. For the hard decoding scenario, the packets are decoded at the receiver and a CRC check is performed. If the CRC passes, the packet is passed up to the network layer. If CRC fails or the packet is lost, the network layer generates an erasure packet indicating an erasure on the channel. For the soft decoding scenario, the packets are decoded at the receiver, but the CRC is bypassed and the soft information for each bit within the packet is used at the network decoder. This will be discussed further in Section IV.

III. ENCODING

Traditionally, a convolutional code is used to code over a block of bits to produce a larger block of bits that contains redundant information. For network coding, we propose to use a convolutional code to code across a block of \( K \) packets to produce a block of \( N \) network coded packets. The convolutional code should be optimized for the desired coding rate, via selection of the actual mother code, mother code rate, and puncturing patterns; and for complexity, via the constraint length and rate of the mother code. The convolutional code may be systematic or non-systematic. In Figure 2, a non-systematic, rate \( \frac{1}{2} \) convolutional network coder with constraint length \( \nu = 7 \) is depicted.
Each input to the convolutional encoder is an entire packet of length $L$. Since linear packet-wise operations, such as addition of packets, can be viewed as several symbol- or bit-wise operations happening in parallel, the process of applying a convolutional code to packets can be interpreted as applying a convolutional code across the $i^{th}$ symbol or bit entries of the all $K$ packets, for $i = 1, \ldots, L$ simultaneously. In this interpretation, there are $L$ encoders which each have an input of $K$ symbols or bits, one from each packet. For the particular rate $1/2$ code shown, the output of each encoder will be each to $2K$ symbols or bits. Putting together the output of all encoders, the output is $2K$ network coded packets. The importance of this idea will be developed further in the following paragraphs.

The output of the convolutional network coder are packets that are linear combinations of the current packet and the previous $\nu - 1$ packets. When performing addition across multiple packets, the bits in the packets are added element-wise. This means that each element, i.e. bit in the output packet, is a linear combination of the element from the current packet and previous $\nu - 1$ packets in the same position. If we think about each packet of length $L$ bits forming a column of an input packet matrix with size $L \times K$, a convolutional network code can be represented as parallel encoding of the bits in each row of the matrix.

For $K$ message packets, which are denoted as $m_i$ for $i = 1, \ldots, K$, of length $L$ bits, a convolutional network code of general rate $K/N$ is modeled as parallel encoders operating over $L$ input blocks of bits, $b_i$ for $i = 1, \ldots, L$, which are made up of the elements $m_{j,i}$ for $j = 1, \ldots, K$, meaning

$$b_i = (m_{1,i}, m_{2,i}, \ldots, m_{K,i}).$$  \hspace{1cm} (1)

This is depicted in Figure 3. If we consider the packets as columns vectors of length $L$, then the message packets form a matrix of size $L \times K$. The network convolutional encoding can then be viewed as encoding each row of the matrix with a convolutional code. The output blocks are of length $N$ and define the corresponding entries in the $N$ network coded packets. If the input packets are not of the same size, the smaller packets can be increased in length to match the size of the largest packet by adding pad bits. These pad bits are typically inserted as zero bits.

Traditionally, the longer the input string of bits, the better the performance of a convolutional code (up to a point); in our case, the length of the input is determined by the number of packets, $K$. Since $K$ packets need to be stored when using a convolutional network code, this is very expensive in terms of memory. To solve this memory problem, we propose a method for reducing the number of packets that must be stored in memory without sacrificing performance. Our solution is to apply the convolutional code both across packets and within the packets, i.e., divide the packets into multiple concatenated blocks of bits and form virtual packets, and then use a convolutional encoder across the virtual packets.

We essentially transform a small number of long packets into a larger number of shorter packets. This is equivalent to reshaping the input packet matrix to have size $L/D \times KD$, where $D$ is defined to be the coding depth of the convolutional network code. For example, a coding depth of $D = 4$ means that each packet is divided into four equal portions to create $4K$ virtual packets of length $L/4$ and the convolutional encoder works across these $4K$ virtual packets. This is depicted in Figure 4. A coding depth of $D = 4$ implies that only half the number of packets are needed to achieve performance similar to coding over $4K$ packets of original length. In summary, for coding depth $D > 1$, the number of packets that needs to be stored in memory decreases as the coding depth increases.

After encoding over the shorter virtual packets, the output
packets must be reshaped back to input a packet matrix consisting of packets of the original length, $L$. With an input packet matrix to the convolutional network code of size $L/D \times KD$, the encoded packet matrix will be of size $L/D \times ND$ for the $K/N$ network code. This encoded packet matrix is reshaped prior to transmission to be of size $L \times N$, or $N$ packets of length $L$. Thus, the process of increasing the coding depth does not alter to overall network coding operation of producing $N$ network coded packets from $K$ original packets. However, the same reshaping process will occur when decoding the packets and, as will be discussed further, a larger coding depth may increase the frequencies of errors seen by the decoder.

IV. DECODING

At the receiver, a traditional Viterbi decoder is used at the network layer. Two types of inputs to the decoder are considered: (1) hard information, (2) soft information.

First, we describe how to design a network decoder that uses hard information as the input. The output of the traditional inner system (PHY and MAC) is either the successfully recovered packet or an erasure, if the packet failed the CRC. We propose the following function, $f$, that maps the success or failure of receiving a network coded packet to reliability information (or log-likelihood ratio information). As an input, $f$ takes a vector of length $L$, which is either a packet of bits, $p$, or a packet erasure, $\epsilon$, and is defined as follows:

$$f(x) = \begin{cases} 0 & x = \epsilon \\ C(2x - 1) & x = p \end{cases}$$

The scaling factor, $C$, is selected to be a large number representing a strong probability of the bits in packet $p$ being correct, since the packet passed CRC. The output of $f$ is a packet of size $L$. The ‘0’ above represents a vector of $L$ zeros, reflecting no confidence in the packet’s bits.

If a coding depth $D > 1$ was used for encoding, the same depth is used for the Viterbi decoder. Thus, the received packet matrix of size $L \times N$ is reshaped to a matrix of size $L/D \times ND$ for the process of applying the Viterbi decoding operation. This reshaping results in no-confidence bits appearing at periodic intervals at the Viterbi decoder. If packet $p_i$ fails the CRC, then the input to the Viterbi decoder has no-confidence bits in every position with index $tN+i$, for $t = 0, \ldots, \frac{KD}{N}$. It is reasonable to assume that the PER for the link is sufficiently low such that only one packet per block of $N$ network coded packets fails CRC. In this case, the period between erasure bits is $N$. To ensure that the Viterbi decoder correctly determines these bits, the no-confidence bits must not be spaced within the minimum distance, $d_{\text{free}}$, length; therefore, the network code should have an associated $d_{\text{free}}$ such that $d_{\text{free}} < N$. Recall that $d_{\text{free}}$ describes the minimal Hamming distance between different encoded sequences.

Next, we describe decoding with soft information. If soft information is passed up from the MAC layer, this is passed directly to the Viterbi decoder. In this case, the CRC information is bypassed entirely and only the soft information is used. Since there are no packet erasures, there are no periodic erasure bits at the Viterbi decoder for $D > 1$, as in the case with hard information used for decoding. Thus, the coding depth does not affect the overall performance.

V. RESULTS

Figure 5 shows the PER results for convolutional network coding with a hard-decision network decoder using various coding depths ($D = 1, 4, 100, 250$). In the simulations, a rate 1/2 convolutional network encoder with constraint length $\nu = 7$ and block diagram shown in Figure 2 was used. At the physical layer (PHY), the same convolutional encoder shown in Figure 2 was used for the channel convolutional encoder and the resulting coded bits were mapped onto a Gray coded 64-QAM constellation. The corresponding free distance for this convolutional code is $d_{\text{free}} = 10$. In all of the cases, the product of the number of packets and coding depth was kept constant, i.e., $KD = 1000$.

For the hard-decision network decoder, Figure 5 shows that there is 0.55 dB coding loss when going from $D = 1$ to $D = 250$; however, the memory storage decreases by a factor of 250, i.e., from storing 1000 packets to storing 4 packets. In addition to the coding loss, there is also a change in the slope of the curves. For $D = 1$, the PER decreases from 1 to $10^{-4}$ in 0.8 dB, whereas for $D = 250$ it takes 1.4 dB, implying that there is a loss in diversity due to the repetitive nature of the erasure pattern seen by the Viterbi decoder.

It is interesting to note the slope of the PER curves for all coding depths relative to the case non-network-coded case. With network coding, the slope of the PERs is significantly greater than for the non-network-coded PER curve. A stronger channel code would give a PER coding gain, but would not increase the steepness of the slope to such a high degree.
For the soft-decision network decoder, the coding gain is much larger, as expected; these results are shown in Figure 6. Comparing the two figures, at a PER = 10⁻³, there is 3.8 dB gain between a soft-decision decoder and a hard-decision decoder when D = 250. At the same PER, there is a 3.3 dB gap when D = 1. As seen in Figure 6, the performance of a soft-decision decoder is independent of the coding depth since there are no erasures and, therefore, no periodic erasure information seen by the Viterbi decoder.

VI. CONCLUSIONS

In this paper, we propose a new type of network codes, namely convolutional network codes, for improving the reliability of a point-to-point wireless channel. This area of research is fairly new and network codes hold much promise for increasing the robustness of wireless networks. We propose combining knowledge of network coding and error correcting codes to come up with convolutional network codes, which were shown to provide a large improvement in PER compared to a system without network coding.

Additionally, we investigated how to reduce the memory required to implement such a network code. The idea of reshaping packets and increasing coding depth was analyzed relative to the final performance of the code. We gave direct insight on how to select parameters for the convolutional code chosen based on the expected erasure patterns at the network decoder that result from using a coding depth D > 1.

The network codes proposed with hard decoding are compatible with existing technologies, meaning that they can be added above the MAC layer to an existing wireless system and increase the reliability of the system without altering how the system operates below the MAC layer. The network codes with soft decoding require some manipulation of the underlying wireless system, with the tradeoff being that a substantial coding gain of several dB can be achieved. Convolutional network codes are thus both theoretically interesting and of practical importance for realization in existing wireless systems.

ACKNOWLEDGMENTS

The authors would like to acknowledge June Chul Roh, Member of Technical Staff at Texas Instruments, for his valuable and insightful comments during the development of the work presented in this paper.

REFERENCES