EC601 Product Design in Electrical and Computer Engineering

BAtch Sparse Code and some related projects

Literature Review

Tianyi Xu

tyx@bu.edu

1.Network Coding

1.1 Basic

The idea of network coding was first introduced in a paper called "Network Information Flow" by Ahlswede *et al.* This concept mainly defines that the coding at a node in a network is network coding.[1]

They further explained this concept in detail. They explained that in the existing network (without network coding), each node in the network functions as a switch. This switch would operate in either of two ways in one operating or functioning, namely, 1) single forwarding (relays information from an input link to an output link), 2) multi-forwarding (replicates information received from an input link and sends it to a certain set of output links)[1]. However, it's not necessary to limit the function of a node to that of a switch. Hence, then try to utilize nodes in the network for not just forwarding information, but also as an encoder. This encoder (node) would receive information from all the input links, encode the information it received, and send information to all the output links. This encoding process is the key part of this network coding concept as it expands the function of one node in the existing network to perform calculation on the information it receives.

1.2 Conditions and Achievement.

Researchers have restrained their conditions to the problem under noiseless communication links, with one information source, with one or more destinations, or in another word, a one source multicast case. Under this constraint, they have obtained a simple characterization of the admissible coding rate region. It reveals that by employing coding at the notes, bandwidth can in general be saved. It points out that the traditional technique for multicasting in a computer network in general is not optimal.

2. Linear Network Coding

2.1 Basic

As described in the previous section, the concept of network coding has been established. The remaining question is how to perform this coding process, or what's the proper way of coding the information received by each node in order to achieve a better information-flow efficiency. Li, *et al.* have proposed in their paper "Linear Network Coding" that linear coding suffices to achieve the optimum, which is the max-flow from the source to each receiving node in a multicast case.[2]

One stricter notion of network coding has also been made in this paper: the notion of network coding refers to coding at the intermediate nodes when information is multicast in a network.[2]

To illustrate the idea of a linear network coding, they started with a so-called 'Butterfly Network', which takes one node as source node and two nodes as sink node/destination node. By performing exclusive-OR to the two bits simultaneously(theoretically) received by the intermediate node and then sending this 'operated' information to the sink notes, a higher throughput has been achieved in this multi-hop network.

In practice, they treat a block of data as a vector over a certain base field and allow a nod to apply a linear transformation to a vector before passing it on to the next node. This process is referred to as linear network coding.

2.2 Conditions and Achievement

Under noiseless communication link case, for source nodes, information is multicast in the multihop fashion where every node can pass on any of its received data to other nodes.
The proved that by linear coding alone, the information rate at which a message reaches each node can achieve the individual max-flow bound. This linearity approach also facilitated the encoding and decoding process of information.[2]
They proposed that further research problems in network coding include 1) multisource network coding problem: code construction when two or more sources are simultaneously multicast in the network, 2) synchronization problem for real-time application.

3. Erasure Code

3.1 Basic

In the existing network configuration with noisy channels, we usually take an error-correcting code approach to make sure the packet received at the receiver side is the same as what the sender holds, if it's not the same, then we would send a feedback message from receiver to sender. This feedback may contain the information of the missing packets, or an acknowledgment for each received packet [4]. With this backwards message, even if the packet is lost, we can still notify the sender and let it retransmit the lost packet. However, this approach is wasting a lot of resources of the channel, decreasing the channel utility. We want to design a forward-error-correction code, in these erasure channels, to avoid these drawbacks. As a result, we have one erasure-correcting code called Reed-Soloman codes, which requires no feedback or almost no feedback.[4]

3.2 Constraints

These Reed-Soloman codes are only practical for small packet/symbol size, since the computational cost is too much for a bigger packet size.

4. Fountain code

As the name implies, the encoder of this Fountain code is generating encoded packets endlessly, like a fountain, and the waterdrops are the encoded packets/symbols. The receiver/intermediate nodes in this fountain network would hold a digital bucket (buffer) to store all the digital drops(packets). The receiver would have a high probability to decode the original information from the digital drop if it received a certain number of encoded packets (usually greater than the original packet)

The major difference/property of a fountain code is its ratelessness, since the number of encoded packets that can be generated form the source/original packets is potentially limitless [4]. This property leads to an advantage that these Fountain codes are simultaneously near-optimal for every erasure channel.

One of the simplest examples of Fountain code would be the Random Linear code (Random Linear Fountain). However, this code has fatal drawbacks. If the number of packets is too large, the computational and storage cost would be tremendous.

5. Batched Spares Code

5.1 Mission and purpose

1) Low encoding complexity in the source node and low decoding complexity in the destination node, 2) constant computational complexity for encoding a packet at an intermediate node and

constant buffer requirement at an intermediate node, 3) small protocol and coefficient vector overhead, 4) high transmission rate.[7]

5.2 Basic

Batched Sparse Code extends fountain code to incorporate with random linear network coding. This random linear network coding is playing the role of an erasure code. More specifically, BATS consist of 2 layers of codes, respectively, the Outer code and the Inner code.
The Outer code is a matrix generalization of a fountain code. This outer code would first create batches from coded packets generated from a subset of the input packets. This process can be regarded with a linear combination of the original packets, by representing a subset of original packets to a generator matrix. This subset of original packets is specified by a degree distribution and the generator matrix is generated randomly. Once this batch is generated by the other code, the following inner code is only performed inside these batches. The inner code applies another linear transformation on each batch and is represented by the linear transfer matrices of the batches. Then in the end/destination nodes, a Belief Propagation has been performed to decode the batches in an effective way, since the inner code preserves the degrees of the batches.[8]

Other related works

LT codes [3]

Raptor Codes [6]

Online Codes [5]

FUN codes [9]

These would not be discussed in detail in this simple review, but literature review of the related papers has been finished individually as a reference to the topics discussed above.

References

[1] Ahlswede, R., Cai, N., & Yeung, R. W. (2000). Network information flow theory. *IEEE Transactions on Information Theory*, *46*(4), 1204–1216. https://doi.org/10.1109/ISIT.1998.708784

[2] Li, S.-Y. R., Yeung, R. W., & Cai, N. (2003). Linear network coding. *IEEE Transactions on Information Theory*, *56*(6), 371–381. https://doi.org/10.1109/TIT.2010.2046215

[3] Luby, M. (2002). LT codes. *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, *January 2002*, 271–280. https://doi.org/10.1109/SFCS.2002.1181950

[4] MacKay, D. J. C. (2005). Fountain codes. *IEE Proceedings - Communications*, *152*(6), 1062 – 1068. https://doi.org/10.1049/ip-com

[5] Maymounkov, P. (2002). Online codes. *Secure Computer Systems Group*.

[6] Shokrollahi, A. (2006). Raptor codes. *IEEE Transactions on Information Theory*, *52*(6), 2551–2567. https://doi.org/10.1201/b11707-21

[7] Yang, S., & Yeung, R. W. (2014). Batched sparse codes. *IEEE Transactions on Information Theory*, *60*(9), 5322–5346. https://doi.org/10.1109/TIT.2014.2334315

[8] Yang, S., Yeung, R. W., Cheung, J. H. F., & Yin, H. H. F. (2014). BATS: Network coding in action. *2014 52nd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2014*, 1204–1211. https://doi.org/10.1109/ALLERTON.2014.7028592

[9] Zhang, H., Sun, K., Huang, Q., Wen, Y., & Wu, D. (2016). FUN Coding: Design and Analysis. *IEEE/ACM Transactions on Networking*, *24*(6), 3340–3353. https://doi.org/10.1109/TNET.2016.2516819