Raptor Packets: A Packet-Centric Approach to Distributed Raptor Code Design

Dejan Vukobratović, Čedomir Stefanović
Department of Power, Electronics and Communication Engineering, University of Novi Sad, Novi Sad, Serbia.
Email: {dejanv, cex}@uns.ac.rs

Miloš Stojaković
Department of Mathematics and Informatics, University of Novi Sad, Novi Sad, Serbia.
Email: smilos@inf.ethz.ch

Vladimir Stanković
Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, UK.
Email: vladimir.stankovic@eee.strath.ac.uk

***Abstract*—In this paper, we address the problem of distributed Raptor code design over information packets located across the network nodes. We propose a novel approach to this problem that consists of generating, encoding and dispersing *Raptor packets* across the network. Unlike recent node-centric proposals, where network nodes are responsible for collecting information packets and performing Raptor encoding, in the proposed packet-centric approach this task is assigned to Raptor packets. In a two-step encoding procedure that corresponds to precoding and LT-coding step of standard Raptor encoding, Raptor packets randomly traverse the network, collect and encode sufficient number of information packets following exactly a given degree distribution, and finish their paths in a random network node. The efficiency of the distributed Raptor coding scheme is confirmed by simulation results, where their performance is demonstrated to approach closely the performance of standard (centralized) Raptor codes.**

I. Introduction

Rateless (or fountain) codes, such as LT codes [1] or Raptor codes [2], are recently proposed capacity-approaching sparse-graph codes universally efficient over erasure channels regardless of their erasure statistics. As opposed to standard fixed-rate block codes, rateless codes do not have specified rate prior to transmission. In other words, at the transmitter side, rateless codes are able to encode a message containing N information packets into a stream of potentially infinite number of encoded packets. At the receiver side, collecting any slightly more than N encoded packets should be sufficient for recovering the original message with high probability using the simple iterative Belief-Propagation (BP) decoding algorithm. Due to their rateless property, efficiency and linear encoding/decoding complexity, state-of-the-art Raptor codes are recently recognized as an attractive solution and proposed in several large-scale content distribution systems.

Rateless codes are usually explored in the context of multi-casting a message from a source node to a number of receiving nodes in the network. In such scenario, rateless encoding is centralized process implemented at the source node where the message to be transmitted is completely available. However, in some networking scenarios such as ad-hoc wireless networks or wireless sensor networks (WSN), the message of interest could be distributed across the network nodes. Due to attractive properties of centralized rateless codes, a simple and efficient rateless coding over the data content distributed across the network nodes became of research interest lately [3]-[7].

In this paper, we propose a simple and efficient solution for distributed Raptor code design. The main idea of the proposed approach is to generate, encode in a distributed fashion and disperse uniformly across the network a sufficient number of encoded packets called *Raptor packets*. Each Raptor packet should share the same properties as if it was generated by the centralized Raptor encoder. Collecting any subset of Raptor packets slightly larger than the size of the distributed network data content is sufficient for successful data recovery with high probability. As opposed to recently proposed node-centric distributed rateless coding schemes, where network nodes are responsible for collecting information packets and performing rateless encoding [5]-[7], the proposed approach is packet-centric as the encoding task is controlled by the Raptor packets themselves. Creating Raptor packets is a two-step procedure that corresponds to precoding and LT-coding step in standard Raptor codes. Both steps apply the same idea, where parity packets (in the first step) or Raptor packets (in the second step), initially assigned with a randomly selected degree from a given degree distribution, randomly traverse the network collecting and encoding into their content a desired number of uniformly selected information packets. Once the chosen degree is encoded, the packet terminates its random walk in a randomly selected node.

The packet-centric concept provides clear advantages: it is simple and easy to implement, and it provides encoding with *exact* degree distributions unlike node-centric methods which approximate the desired degree distributions. It is applicable in many networking scenarios such as data gathering [5], data persistence [4] or distributed network storage [3][6][7]. The efficiency of the proposed scheme is shown to approach closely the performance of centralized Raptor codes.

II. Background

A. Raptor Codes

LT codes [1] are the first practical capacity-approaching rateless codes. LT encoding is simple process where, for each encoded packet, a degree d is sampled from a degree distribution $\Omega(d)$, and d out of N information packets of the

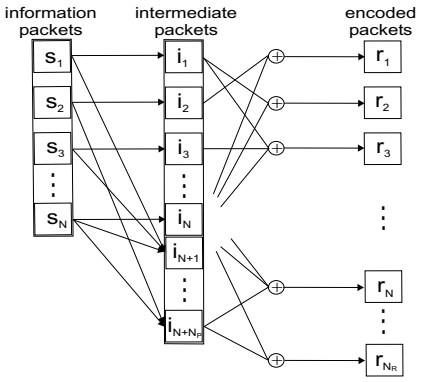


Fig. 1. Raptor code graph consisting of LDPC precode and LT code graph.

source message are uniformly selected and XOR-ed to produce the encoded packet. In [1], Robust Soliton degree distribution $\Omega_{RS}(d)$ is designed, which enables capacity-achieving performance of LT codes with the iterative BP decoder. Using LT codes, the receiver is able to decode the source message with any $N+O(\sqrt{N}\ln^2(N/\delta))$ received encoded packets with probability $1-\delta$. However, as the average degree of $\Omega_{RS}(d)$ scales as $O(\ln(N/\delta))$, the average LT encoding/decoding complexity grows as $O(N\ln(N/\delta))$.

Raptor codes [2] are capacity-approaching rateless codes with linear encoding/decoding complexity. They consist of the high-rate LDPC (Low Density Parity Check) precode concatenated with the LT code defined by a weakened, constant average degree distribution $\Omega_R(d)$. The idea is first to recover a constant, close to one fraction of intermediate packets (i.e, the precode LDPC codeword) from received encoded packets using the weakened linear encoding/decoding complexity LT code, and then to recover all of the information packets by exploiting the high-rate linear encoding/decoding complexity LDPC precode. As the precode, one can apply any LDPC code design method providing good high-rate LDPC codes. Examples are left-regular right-Poisson LDPC codes proposed in [2] or irregular repeat-accumulate (IRA) LDPC codes [8]. In the LT coding phase, different options for the degree distribution $\Omega_R(d)$ are discussed in [2], both for asymptotic and finite-length Raptor code design. The structure of Raptor codes that consist of the LDPC precode and the LT code graph is illustrated in Fig. 1. We assume the iterative BP decoding of Raptor codes that first operates on the LT part, and then on the LDPC precode part of the code graph.

B. Random Walks on Graphs and Uniform Node Sampling

Uniform sampling of network nodes can be simply and efficiently performed using random walks on graphs. We provide a short background on the topic, as it is important part of distributed Raptor coding scheme using Raptor packets.

Let $\mathcal{G}(V,E)$ be a connected undirected graph with the vertex set V, $|V|=N$, and the edge set $E\subseteq V^2$, $|E|=m$. The node i in the graph is connected with the node j iff $(i,j)\in E$. The set $\mathcal{N}(i)$ of all neighbors of node i is the set of all nodes in V connected with i, where $|\mathcal{N}(i)|=d(i)$ is called the degree of the node i. Random walk on $\mathcal{G}(V,E)$ is a sequence of nodes starting from any node, such that the next node j in the sequence is selected from $\mathcal{N}(i)$ with probability p_{ij}, where i is the previous node in the sequence. Random walk on graph can be modeled as a Markov Chain (MC), where the states of MC correspond to the graph vertices, and the transition probability matrix of MC, $\mathbf{P}=[p_{ij}]$, is defined by next-hop probabilities p_{ij}. From the theory of MC, it is well-known that if $\mathcal{G}(V,E)$ is undirected, aperiodic and connected, the corresponding MC is irreducible, homogeneous and aperiodic and converges to the stationary set of state probabilities $\boldsymbol{\pi}=(\pi_1,\pi_2,\ldots,\pi_N)$ that satisfy $\boldsymbol{\pi}=\boldsymbol{\pi}\mathbf{P}$. If a random walk selects any neighbor j from the set $\mathcal{N}(i)$ of the node i equally likely, i.e., with probability $p_{ij}=1/d(i)$, it is called normal random walk (NRW). Although easy to implement, NRW will converge to the stationary distribution $\boldsymbol{\pi}$ where $\pi_i=d(i)/2m$, which is uniform only for regular graphs.

For the problem of uniform node sampling in any (irregular) graphs, the transition matrix $\mathbf{P}$ has to be designed to provide uniform stationary distribution $\boldsymbol{\pi}_U$, where $\pi_i=1/N, 1\leq i\leq N$. It is easy to show that the matrix $\mathbf{P}$ provides the stationary distribution $\boldsymbol{\pi}_U$ iff it has non-negative entries ($\mathbf{P}>0$), is symmetric ($\mathbf{P}=\mathbf{P}^T$) and doubly-stochastic (any row/column sum up to one). Two simple and popular heuristic methods for designing $\mathbf{P}$ that satisfy the above conditions are maximum-degree (MD) and Metropolis-Hastings (MH) algorithm [9][10].

The length of the random walk needed to approximately reach the stationary distribution is called the mixing time of the corresponding MC [11][12]. Informally, mixing time $\tau=\tau(\epsilon)$ is the number of steps of the random walk required for the probability of visiting any node i after $\tau(\epsilon)$ steps, $\pi_i^{\tau(\epsilon)}$, is ϵ-close to the stationary probability π_i. Mixing time of MC is related to the second largest eigenvalue $|\lambda_2|$ of the corresponding probability transition matrix $\mathbf{P}$ and scales as $\tau=O(\log N/(1-|\lambda_2|))$.

C. Virtual Graph Approach to Uniform Packet Sampling

If each network node contains exactly one data packet, uniform sampling of data packets and network nodes are equivalent tasks. However, the number of data packets per network node need not be one, or equal for each node. Uniform sampling of data packets in such a scenario is recently addressed in [13] using the concept of virtual graphs, where the main idea is to apply random walk on the virtual graph derived from the original network graph. Virtual nodes of the virtual graph correspond to data packets, and the connectivity between the virtual nodes is derived locally, using simple information exchange among neighboring network nodes. After establishing the virtual graph, uniform data packet sampling proceeds as the previously described uniform node sampling: using random walks on the virtual graph. Due to space constraints, we skip the details on the virtual graph approach and refer the interested reader to [13]. However, to complement this short introduction, we provide additional implementation details on virtual graph design in the following section.

III. Distributed Raptor Coding Using Raptor Packets

We represent a network containing N network nodes by an undirected connected graph $\mathcal{G}(V, E)$. A *network message* containing N equal-length information packets is distributed across all network nodes so that each node possess exactly one information packet. Our goal is to design a distributed Raptor coding scheme that produces desired number N_R of encoded packets, called Raptor packets, distributed across the network. Each Raptor packet should share the same properties as if it was generated by the centralized Raptor encoder. Raptor packets are generated from the network message in two phases: the precoding and the LT coding phase, where both phases apply the same packet-centric approach to packet encoding. In other words, each Raptor packet is initially associated with a randomly selected degree from a given degree distribution and randomly traverses the network collecting information packets. When a given degree is reached, the Raptor packet is stored in a random network node. After the distributed Raptor packet encoding is finished, retrieving any slightly more than N Raptor packets (as $N \to \infty$) from any set of network nodes should be sufficient for the network message recovery with high probability. In the following, we provide a detailed description of the Raptor packet approach.

A. Precoding Phase

The goal of the precoding phase is to create and distribute across the network N_P *parity packets* corresponding to redundant packets of a high-rate LDPC precode. The parity packets are created from N information packets of the network message, and both information and parity packets represent a codeword of the LDPC precode. Hence the first problem we need to solve is the design of distributed systematic rate R LDPC precode that outputs $N_P = ((1-R)/R)N$ parity packets. After the precoding phase is finished and N_P parity packets are dispersed across the network, a total of N information packets and N_P parity packets are equally treated as a new, precoded network message to be encoded in the second, LT coding phase. The precoded network message contains $N_I = N + N_P = N/R$ *intermediate packets*.

In this paper, we apply a simple class of distributed precodes called Low-Density Generator Matrix (LDGM) codes, where LDGM parity packets are created by combining subsets of information packets. Information and parity packet degrees are described by node-oriented left and right degree distributions $\Lambda(x)$ and $P(x)$ respectively[1]. Parity packets of distributed LDGM codes are created using the same simple packet-centric approach, reused later in creating Raptor packets in the LT coding phase. With this approach, we obtain $P(x)$ exactly, whereas $\Lambda(x)$ can be designed sufficiently close to the desired distribution.

Creating parity packets proceeds in two steps: initialization and encoding step.

[1]In Fig. 1, $\Lambda(x)$ and $P(x)$ describe the subgraph between information packets $\{s_1, s_2, \ldots, s_N\}$ and parity packets $\{i_{N+1}, i_{N+2}, \ldots, i_{N+N_P}\}$.

Fig. 2. Parity/Raptor packet header fields.

Initializing Parity Packets: In this step, each network node creates a parity packet with probability $(1-R)/R$. This way, approximately $N_P = ((1-R)/R)N$ parity packets are created in N_P network nodes. Each parity packet is initialized as a copy of the information packet from the same node and its header fields (Fig. 2) are initialized as follows. To each parity packet, a random degree d_r from $P(x)$ is assigned. As one information packet is already included in the parity packet by initialization, the degree counter header field, representing the remaining degree to be collected, is set to $d_r - 1$. Mixing time counter field is initialized with the selected value τ, which is a global constant. Finally, the ID of the information packet used to initialize the parity packet is placed in the header field containing IDs of all information packets that are combined into the parity packet. Additionally, each network node randomly selects and stores its own node degree counter value d_l drawn from $\Lambda(x)$. The node degree counter will be used to enforce a given $\Lambda(x)$ during the precoding phase.

Encoding Parity Packets: Upon initialization, N_P parity packets independently perform the encoding step. The goal of each parity packet is to add to its content the remaining $d_r - 1$ information packets selected uniformly at random. Selecting an information packet uniformly at random is equivalent to selecting a network node uniformly at random. Parity packets achieve this task by performing random walks on network graph as described in Section IIB. This means that, after visiting one node, the parity packet randomly selects the next node to visit from the set of the previous node neighbors. The probability p_{ij} of selecting the node j from $\mathcal{N}(i)$ for each node i are obtained locally by each network node.

Along their random walk, parity packets are processed by each network node. The network node processing of the parity packet is simple and is described as follows. If the mixing time counter $\tau > 0$, it is decremented by one and the parity packet is forwarded to the next random hop. If $\tau = 0$ and the node degree counter $d_l > 0$, the node XOR-s its own information packet to the content of the parity packet, updates the parity packet header fields: decreases the degree counter, adds information packet ID and resets the mixing time counter to its initial value, and decreases the node degree counter by one[2]. If the mixing time counter is equal to zero and $d_l = 0$, the parity packet continues its random walk searching for the neighboring node with $d_l > 0$. If such a node is not found within limited number of hops, the information packet of the last visited node is encoded into the parity packet content. With this rule, the left degree distribution $\Lambda(x)$ is closely approximated. After d_r uniformly selected information

[2]An exception to the rule occurs only if the parity packet already contains the information packet in its XOR-sum. In that case, the parity packet continues its random walk until it reaches the first network node containing information packet which has not yet contributed to its encoded content.

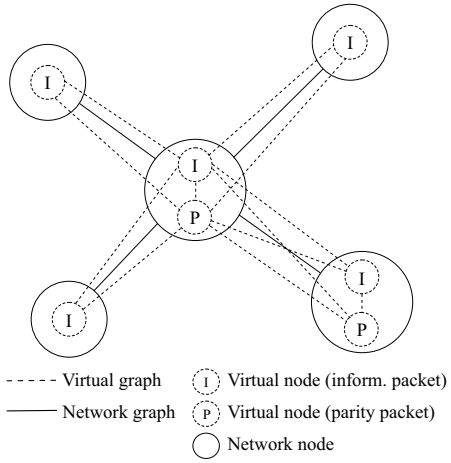


Fig. 3. Network graph and Virtual graph.

packets are encoded into the parity packet, its encoding phase is completed.

For simplicity of the following LT coding phase, we limit the number of parity packets per network node to be equal to one. Therefore, if parity packet ends the encoding phase in the network node containing another parity packet, it continues its random walk until the first node that has no parity packets. With this constraint, after the precoding phase, there are N_P network nodes, each containing one information and one parity packet, and $N - N_P$ network nodes, each containing one information packet only.

B. LT Coding Phase

In the LT coding phase, $N_R = bN_I$ Raptor packets are created from the set of $N_I = N + N_P$ intermediate packets. The number of Raptor packets b created per intermediate packet is used to control the total number of Raptor packets available in the network. Creating Raptor packets proceeds in three steps: initialization, encoding and dispersing step.

Initializing Raptor Packets: Each network node creates initial Raptor packets as b copies of each of its own intermediate packet. In other words, if a node contains one information packet, it will initialize b Raptor packets; if it contains one information and one parity packet, it will initialize $2b$ Raptor packets. The degree d from a selected degree distribution $\Omega_R(d)$ is assigned independently and randomly to each Raptor packet. The degree counter of value $d-1$, representing the remaining degree to be collected, is placed in the Raptor packet header. Additionally, Raptor packet header fields are initialized with the mixing time counter and the ID of the intermediate packet used to initialize the Raptor packet (Fig. 2).

Encoding Raptor Packets: After initialization, bN_I Raptor packets start their encoding step, where each Raptor packet adds to its content the remaining $d-1$ intermediate packets selected uniformly at random. As intermediate packets are not in one-to-one correspondence with network nodes, Raptor packets perform random walk across the virtual graph created on intermediate packets, as described in Section IIC. The construction of the virtual graph is simplified as each network node contains either one or two intermediate packets. The rule for establishing edges in the virtual graph is that each virtual node connects to all the virtual nodes in its network node and in the neighboring network nodes, as illustrated in Fig. 3.

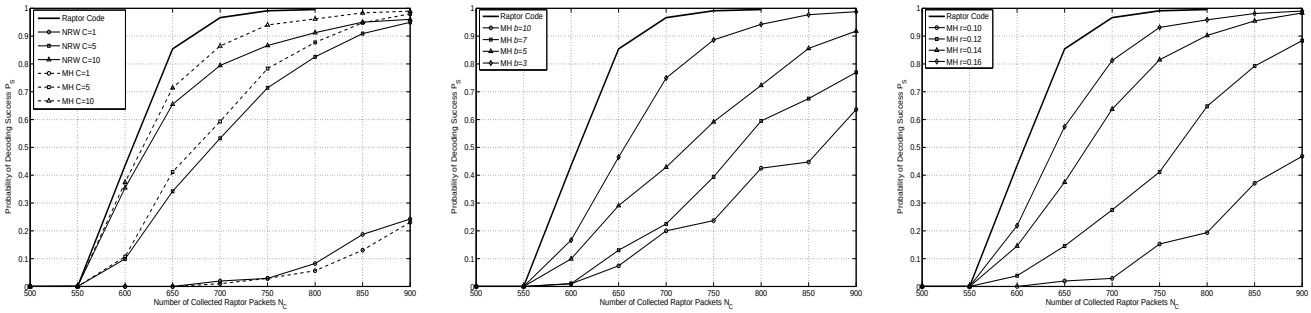
The processing of Raptor packets in virtual nodes is the same as processing parity packets in the precoding phase. If the mixing time counter $\tau > 0$, the virtual node decreases the mixing time counter by one and forwards the Raptor packet to the next random virtual node. Otherwise, if $\tau = 0$, the virtual node XOR-s its intermediate packet to the Raptor packet content and updates the Raptor packet header (decreases the degree counter, adds the intermediate packet ID and resets the mixing time counter to the initial value). Random Raptor packet forwarding across the virtual graph is determined by the probability transitions generated locally by each node. After d uniformly selected intermediate packets are encoded into the Raptor packet, the encoding phase is completed.

Dispersing Raptor Packets: After the encoding step, the Raptor packet should be placed in a random network node. This is achieved by forwarding the Raptor packet for another τ hops across the network graph until it finds its final random position in the network. After dispersing, expected number of Raptor packets per network node is equal to bN_I/N.

IV. Simulation Results

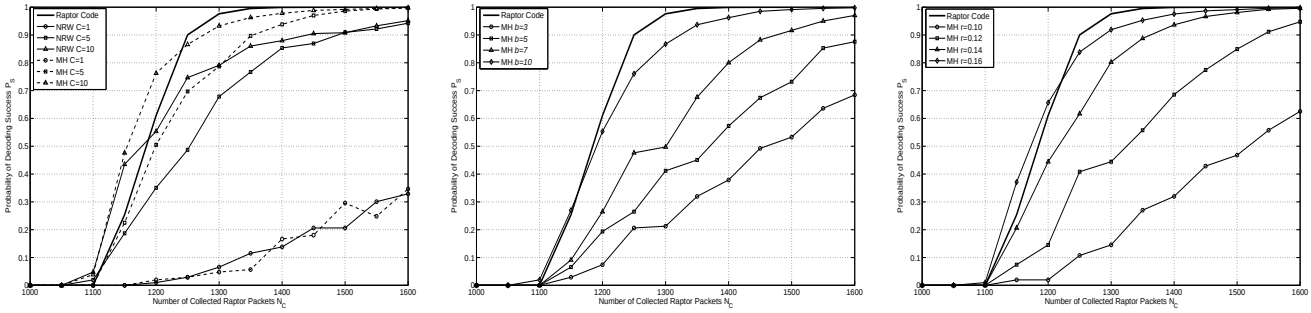
In this section, we analyze the performance of the proposed scheme and compare it with the centralized Raptor codes. For the network model, we assume a random geometric graph $\mathcal{G}(N, r)$ model typical for wireless ad-hoc or sensor networks, where N nodes are uniformly distributed over the unit square area and a node can reliably communicate only with the nodes in its transmission range r. In each simulation run, N_P parity packets are created in the precoding phase, after which $b(N + N_P) = bN_I$ Raptor packets are produced and dispersed across the network during the LT coding phase. In the precoding phase, we apply a distributed version of $R = 0.95$ regular $(4, 76)$ LDGM precode with the left degree $d_l = 4$ and the right degree $d_r = 76$ [14]. In the LT coding phase, weakened LT code degree distribution of maximum degree $d_{max} = 66$ proposed for finite-length Raptor code design is used [2]. The random walk transition probability matrix $\mathbf{P}$ is designed using NRW and MH algorithms. Apart from N and r, the major simulation parameters are the number of Raptor packets b created per intermediate packet, and the mixing time constant C such that $\tau = \lceil C \log N \rceil$. Finally, after Raptor packet dispersion, we assume existence of the mobile collector that collects Raptor packets by performing random walk across the network starting from a randomly selected node. At each node, the collector queries all Raptor packets, and once it collects N_C packets, the iterative BP decoding implemented at the collector is activated.

In Figs. 4 and 5, we present the system performance measured by the probability of decoding success P_S as a function of the number of collected Raptor packets N_C for $N = 500$ and $N = 1000$, respectively. Figs. 4(a) and 5(a)



(a) $N = 500, r = 0.12$, $b = 3$, variable C. (b) $N = 500, r = 0.15$, $C = 3$, variable b. (c) $N = 500$, $b = 3$, $C = 3$, variable r.

Fig. 4. Probability of decoding success P_S vs. number of collected Raptor packets N_C for $N = 500$.



(a) $N = 1000, r = 0.12$, $b = 3$, variable C. (b) $N = 1000, r = 0.15$, $C = 3$, variable b. (c) $N = 1000$, $b = 3$, $C = 3$, variable r.

Fig. 5. Probability of decoding success P_S vs. number of collected Raptor packets N_C for $N = 1000$.

demonstrate that the Raptor packet scheme converges to the centralized Raptor code performance with C for a fixed value $b = 3$, and performs better if MH algorithm is applied. Note that by keeping both b and C small, the total network energy consumption measured by the average number of hops of Raptor packets is kept low. Figs. 4(b) and 5(b) track the system performance for variable b while $C = 3$. Increasing b deteriorates the system performance due to the correlation between the content of Raptor packets collected in the same node or the same part of the network, which increases for smaller C. Figs. 4(c) and 5(c) illustrate the system performance dependence on the node range r. By increasing r, the network connectivity increases and random walk sampling approaches the uniform distribution for the fixed mixing time constant C. However, as r becomes large, the system performance saturates. In order to obtain the desirable system performance, one can balance the choice of parameters r and C, where the optimal trade-off is application specific.

V. Conclusion

A novel distributed Raptor coding scheme based on creating and distributing Raptor packets across the network is introduced. The proposed approach is simple, results in exact Raptor degree distributions, and performs closely to the performance of centralized Raptor codes.

References

[1] M. Luby, "LT Codes," *Proc. of the 43rd Annual IEEE Symp. Foundations of Computer Science (FOCS)*, Vancouver, Canada, November 2002.

[2] A. Shokrollahi, "Raptor Codes," *IEEE Trans. on Information Theory*, vol. 52, No. 6, pp. 2551–2567, June 2006.

[3] A. Dimakis, V. Prabhakaran, K. Ramchandran, "Distributed Fountain Codes for Networked Storage," *Proc. IEEE ICASSP 2006*, 2006.

[4] A. Kamra, J. Feldman, V. Misra and D. Rubenstein, "Growth Codes: Maximizing Sensor Network Data Persistence," *Proc. ACM SIGCOMM 2006*, Pisa, Italy, 2006.

[5] Y. Lin, B. Liang and B. Li, "Data Persistance in Large-Scale Sensor Networks with Decentralized Fountain Codes," *Proc. IEEE INFOCOM 2007*, Anchorage, AL, USA, 2007.

[6] S. Aly, Z. Kong, and E. Soljanin, "Fountain Codes Based Distributed Storage Algorithms for Large-Scale Wireless Sensor Networks," *Proc. IEEE/ACM IPSN 2008*, S. Louis, MO, USA, 2008.

[7] S. Aly, Z. Kong, and E. Soljanin, "Raptor Codes Based Distributed Storage Algorithms for Wireless Sensor Networks," *Proc. IEEE ISIT 2008*, Toronto, Canada, 2008.

[8] H. Jin, A. Khandekar and R. McEliece, "*Irregular Repeat-Accumulate Codes*," Proc. of Turbo-Coding Symposium, Brest, France, 2000.

[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equations of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, pp. 1087–1092, 1953.

[10] S. Boyd, P. Diaconis, and L. Xiao, "Fastest Mixing Markov Chain on a Graph," *SIAM Review, problems and techniques section*, vol. 46(4), pp. 667–689, December 2004.

[11] L. Lovasz, Random walks on graphs: A survey, *in: Combinatorics, Paul Erdos is Eighty*, Vol. 2, Keszthely, 1993, in: Bolyai Soc. Math. Stud., vol. 2, Janos Bolyai Math. Soc., Budapest, 1993, pp. 1–46.

[12] D. Aldous, J. Fill, "Reversible Markov chains and random walks on graphs," unpublished, http://stat-www.berkeley.edu/users/aldous/RWG/book.html, 2008.

[13] S. Datta and H. Kargupta, "Uniform Data Sampling from a Peer-to-Peer Network," *Proc. IEEE ICDCS 2007*, Toronto, Canada, June 2007.

[14] J. Garcia-Frias and W. Zhong, "Approaching Shannon Performance by Iterative Decoding of Linear Codes with Low-Density Generator Matrix," *IEEE Comm. Letters*, vol. 7, No. 6, pp. 266–268, June 2003.