UEP Network Coding for Scalable Data

Apirath Limmanee and Werner Henkel
Jacobs University
EECS, TrSys, Campus Ring 1
28759 Bremen, Germany
Email: {a.limmanee, w.henkel}@jacobs-university.de

***Abstract*—We analyze the effect of erasures on the recovery of scalable data when linear network coding is applied. We find that global encoding kernels (GEKs) describing linear network codes have different levels of built-in unequal-erasure-protecting (UEP) capability, allowing us to better protect the high-priority data when GEKs are wisely assigned. A sub-optimal assignment strategy is suggested to achieve fairness and optimality.**

I. Introduction

It is proven by Ahlswede et al. [2] that, the maximum data flow from the source to each destination in a multicast session can only be guaranteed if network coding is applied. Later, Li et al. [3] proved that a simple class of network coding, called linear network coding, suffices to achieve the maximum flow.

Most papers on network coding model networks as graphs in which each node can code data symbols from incoming edges together and send the resulting coded symbols to each outgoing edge. Each edge usually represents an erasure-free channel with a unit capacity [4] [6] [7] [8].

We follow the same model with one exception: Each edge represents the data transmission rate of one symbol per unit time in a binary erasure channel (BEC), i.e., the edge capacity is reduced from 1 to $1-p$, if p denotes the erasure probability. We discuss this in details in the next section.

Several papers, such as [9], [10], and [11], study error-correcting codes for erasure and error protection. Although the concept can be useful for unequal erasure protection (UEP) of scalable data, it is not included in this work, which emphasizes the effect of global encoding kernels (GEKs) of linear network codes on the recovery of scalable data.

Scalable video and image data consists of several scales of data. Smaller scales represent fine details added to larger ones. Scalable data therefore favours unequal erasure protection (UEP), sometimes mentioned as unequal loss protection (ULP) [1], such that the parts of data with higher priority are better protected against erasures.

We define some related terms and concepts of scalable data in section III, which allows us to give a mathematical expression of the expected quality of the recovered scalable data in the presence of erasures. In section IV, we show that linear network codes have built-in UEP mechanisms affecting the quality of the recovered scalable data. Section V gives an example of a UEP network codes assignment problem, which can be solved by the strategy suggested in Section VI. The last section discusses the result and concludes.

II. The Edge-Disjoint Path Model with Binary Erasure Channels for Network Coding

Figure 1 shows network coding in a network that is usually called the butterfly network, where A aims to multicast two binary symbols b_1 and b_2 to D and E. We can see that the node F encodes b_1 and b_2 together to achieve the multicast rate of 2 bits per unit time, if each edge represents the capacity of one bit per unit time, D receives b_1 from the path ABD and can recover b_2 from the symbol $b_1 \oplus b_2$ from $ACFGD$, whereas E receives b_2 from the path ACE and recovers b_1 from the symbol $b_1 \oplus b_2$ from $ABFGD$. Had network coding not been there, only either b_1 or b_2 would have been able to pass the bottleneck FG in one unit time, i.e., one receiver would have been unable to use one of its possible transmission paths.

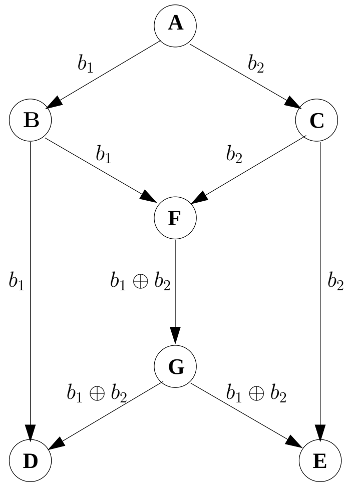


Fig. 1. Network coding in a butterfly network

By means of network coding, all receivers can use all of their possible paths at the same time. In general cases, the multicast rate of ω suggests the existence of ω non-intersecting paths from the source to each sink, which are called edge-disjoint paths by Jaggi et al. [7], although the paths destined to different receivers may share some edges.

If each edge is modeled as a binary erasure channel (BEC), the erasure probability of an edge-disjoint path can be computed from erasure probabilities of all the edges from the source to the sink. The loss of a symbol in each edge-disjoint path does not affect the recovery in another.

Let $p_{i,j,k}$ represents the erasure probability of the k^{th} edge belonging to the j^{th} path of the i^{th} sink. The overall erasure probability of data symbols from the j^{th} path belonging to the i^{th} sink, denoted by $P_{e,ij}$, becomes

$$P_{e,ij} = 1 - \prod_{k=1}^{|E(i,j)|} (1 - p_{i,j,k}), \tag{1}$$

where $|E(i,j)|$ denotes the number of edges in the j^{th} path of the i^{th} sink.

III. The Expected Quality of Received Scalable Data in the Presence of Erasures

Let us consider Fig. 1 again and see what will happen if one symbol is received at each sink whereas another one is erased. At D, if $b_1 \oplus b_2$ is erased, D still obtains b_1. But if b_1 is erased, D obtains neither b_1 nor b_2. This means, for D, b_1 is better protected than b_2. On the other hand, for E, b_2 is better protected. In case b_1 and b_2 are equally important, the network coding is fair. However, if b_1 is more important than b_2 and the erasure probability of each symbol is the same, D is in favor because its more-important symbol is better protected. Thus, to achieve fairness and optimality, the network encoding function of each edge-disjoint path should be carefully chosen.

Before quantitatively expressing the expected quality of the received scalable data when erasures are present, some related terms are defined.

We formulate, in Definition 1, the mapping of scalable data into a scalable message in which each successive symbol adds more details to the data. After explaining the term dependency level in Definition 2, we define an ordered set of scalable data that can be mapped into an ordered scalable message in Definition 3, which possesses an interesting practical property: The incremental quality obtained by each recovered symbol in the message is in a decreasing order. The ordered scalable message thus always prefers more erasure protection in the preceding symbol than the subsequent one.

Definition 1: For any $\mathcal{S} = \{s_1, s_2, ..., s_\omega\}$, representing a set of scalable data with progressively increasing quality from s_1 to s_ω, the i^{th} element s_i can be mapped into a prefix vector $\mathbf{P}_i = [m_1, m_2, ..., m_i]$ of a scalable message $\mathbf{M}= [m_1, m_2, ..., m_\omega]$, which is an ω-dimensional row vector of a finite field $\mathbb{F}$.

$$s_i \longmapsto [m_1, m_2, ..., m_i] \tag{2}$$

Definition 2: A symbol m_k belonging to the scalable message $\mathbf{M} = [m_1, m_2, ..., m_\omega]$ has a dependency level of $\lambda(m_k) = j$ if its significance depends on the successful recovery of the symbols having the dependency level of $j-1$ but not on those with larger dependency level. A symbol of which significance does not depend on any symbol has the dependency level of 1.

Definition 3: Let $\mathbf{Q}(\mathcal{S}) = [q(s_1), q(s_2), ..., q(s_\omega)]$ be a functional vector, of which each element $q(s_i)$, $1 \le i \le \omega$ is a non-negative real number representing the quality of the scalable data s_i in the set $\mathcal{S}$ introduced in Definition 1. Denoting the quality increment vector of $\mathcal{S}$ by $\Delta\mathbf{Q}(\mathcal{S})$ such that

$$\begin{aligned} \Delta\mathbf{Q} &= [\Delta_1, \Delta_2, ..., \Delta_\omega] &(3)\\ &= [q(s_1) - q(s_0), q(s_2) - q(s_1), ... \\ &\quad ..., q(s_\omega) - q(s_{\omega-1})], &(4) \end{aligned}$$

where $q(s_0) = 0$, the set of scalable data $\mathcal{S}$ is said to be ordered if and only if,

$$\lambda(m_u) \ge \lambda(m_v), \text{for } 1 \le u < v \le \omega, \tag{5}$$

where $\lambda(m_j)$ denotes the dependency level of the symbol m_j in the message $\mathbf{M} = [m_1, m_2, ..., m_\omega]$ representing $\mathcal{S}$ and

$$\Delta_i \le \Delta_{i-1}, 1 < i \le \omega. \tag{6}$$

The message $\mathbf{M}$ corresponding to $\mathcal{S}$ is called an ordered scalable message.

From Definition 3, if the prefix $\mathbf{P}_{j-1}$ in the ordered scalable message M has already been successfully recovered, the recovery of the symbol m_j will increase the quality by Δ_j. Therefore, the expected quality $E[Q_i^{\mathcal{S}}]$ of the ordered scalable data recovered at the i^{th} sink is given by

$$\begin{aligned} E[Q_i^{\mathcal{S}}] &= \sum_{j=1}^{\omega} \left[\prod_{l=1}^{j} \varrho_{i,l}\right] \cdot \Delta_j &(7)\\ &= \sum_{j=1}^{\omega} \rho_{i,j} \cdot \Delta_j, &(8) \end{aligned}$$

where $\varrho_{i,l}$ and $\rho_{i,j}$ represent the probabilities that the symbol m_l and the prefix $\mathbf{P}_j$ are recovered at the sink i, respectively. From (8), the quality improves if the term $\rho_{i,j}$, becomes larger, especially for a small j implying a large Δ_j. This reaffirms the essence of UEP, which is to better protect the high-priority preamble. $\rho_{i,j}$ depends on the transmission channels and our network codes, which will be investigated in the next section.

IV. UEP Levels of Global Encoding Kernels for Linear Network Codes

For a network that employs linear network coding, each of its edges in the graphical model, such as Fig. 1, is used to transmit the linear combination of the source symbols. This linear combination can either be represented locally as a linear function of symbols from adjacent edges, which is called "the local encoding mapping", or globally as a linear function of source symbols, which is called "the global encoding mapping" [6]. The global encoding mapping is described by a vector called "the global encoding kernel (GEK)," which is introduced in Definition 4.

Definition 4: Let $\mathbb{F}$ be a finite field, ω a positive integer, and the ω-dimensional, $\mathbb{F}$-valued vector $\mathbf{M}$ the message generated by the source node $\mathcal{S}$. A function $f_e(\mathbf{M})$ of the edge e is said to be a linear global encoding mapping if there exists an ω-dimensional $\mathbb{F}$-valued column vector $\mathbf{f}_e$ such that

$$f_e(\mathbf{M}) = \mathbf{M} \cdot \mathbf{f}_e. \qquad (9)$$

$\mathbf{f}_e$ is called the global encoding kernel [6].

We only consider the problem of assigning, given some local constraints, a suitable global encoding kernel for each edge, in order to optimize the expected quality in (8), since the local encoding mapping can be easily derived thereafter.

To relate the $\rho_{i,j}$ to the network codes, we firstly define the UEP level of a global encoding mapping as follows.

Definition 5: For a scalable message $\mathbf{M}$, which is an ω-dimensional row vector of a finite field $\mathbb{F}$, a global encoding mapping $\gamma_i(\mathbf{M})$ is of the i^{th} UEP level, $0 < i \leq \omega$, if there exists an ω-dimensional, $\mathbb{F}$-valued column vector $\mathbf{C}_i = [c_1, c_2, ..., c_i, 0, 0, ..., 0]^T$, $c_i \neq 0$, such that

$$\gamma_i(\mathbf{M}) = \mathbf{M} \cdot \mathbf{C}_i = \sum_{j=1}^{i} c_j \cdot m_j. \qquad (10)$$

$\mathbf{C}_i$ is then called an i^{th}-UEP-level global encoding kernel.

Now, let us consider an ordered scalable message $\mathbf{M} = [m_1, m_2, m_3]$, where m_1, m_2, and m_3 are binary symbols of the first, second, and third dependency level, respectively. According to Definition 5, there exist seven possible GEKs, one of the first UEP level, two of the second level, and four of the third level, as shown in Table I.

TABLE I
GEKS, THEIR UEP LEVELS, AND THE RESULTING NETWORK-CODED SYMBOLS

UEP Levels	GEKs	Resulting Network-Coded Symbols
1	$[1\ 0\ 0]^T$	m_1
2	$[0\ 1\ 0]^T$	m_2
	$[1\ 1\ 0]^T$	$m_1 + m_2$
3	$[0\ 0\ 1]^T$	m_3
	$[1\ 0\ 1]^T$	$m_1 + m_3$
	$[0\ 1\ 1]^T$	$m_2 + m_3$
	$[1\ 1\ 1]^T$	$m_1 + m_2 + m_3$

From Table I, the prefix $\mathbf{P}_2 = [m_1, m_2]$ can be recovered either from any two symbols from the levels 1 and 2 or from any three symbols from the level 3. For an arbitrary scalable message $\mathbf{M}= [m_1, m_2, ..., m_\omega]$, we can state as a general rule that, in order to recover the prefix $\mathbf{P}_i$, we need either i network coded-symbols belonging to i linearly independent GEKs of which UEP levels do not exceed i, or more than i symbols in case some UEP levels of GEKs exceed i.

Reconsidering eq. (8), assuming that the k^{th} edge-disjoint path of the sink i is used to transmit the k^{th} network-coded symbol to the sink i, the parameter $\rho_{i,j}$, which is the probability that the prefix $\mathbf{P}_j$ is recovered at the sink i, can be written as

$$\rho_{i,j} = \prod_{k=1}^{\omega} \mu_{j,k} \cdot [1 - P_{e,ik}], \qquad (11)$$

where $P_{e,ik}$ denotes the erasure probability of the path k used to transmit the k^{th} network-coded symbol to the sink i. $\mu_{j,k} = 1$ if the k^{th} network-coded symbol is needed to recover the prefix $\mathbf{P}_j$. Otherwise, $\mu_{j,k} = 0$.

In order to make $\rho_{i,j}$ large for small j to satisfy the UEP requirements and optimize the scalable data quality, as mentioned in Section III, we first sort the erasure probability $P_{e,ik}$ such that $P_{e,ik} \leq P_{e,ir}$ for any $1 \leq k < r \leq \omega$. Then we find that the ideal strategy is to allocate a first-UEP-level GEK to the path with the index $k = 1$, such that we only need the path with the lowest erasure probability to recover the most important prefix $\mathbf{P}_1$. In this case, $\rho_{i,1} = 1 - P_{e,i1}$, which is the highest possible $\rho_{i,1}$. Next, we allocate a second-UEP level GEK to the path with next-to-the-lowest erasure probability, i.e., $k = 2$, such that $\rho_{i,2} = (1 - P_{e,i1})(1 - P_{e,i2})$, which is the highest possible $\rho_{i,2}$. We then keep allocating a q^{th} level GEK to the path with the index $k = q$ until reaching the last path.

In practical multicast, however, it may be impossible to apply the ideal strategy to all receivers due to conflicts among them and some side constraints. This problem is discussed in the next section.

V. An Example of UEP Network Coding Problems for Multicasts

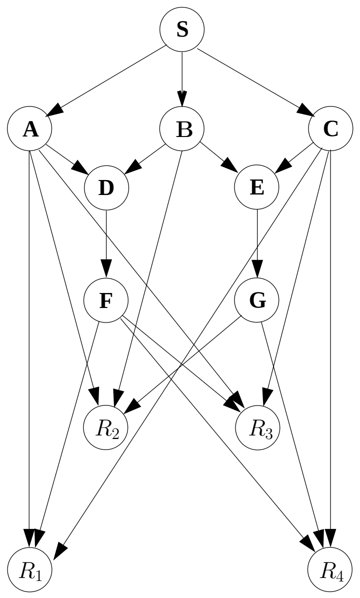


Fig. 2. A network example with a scalable data multicast

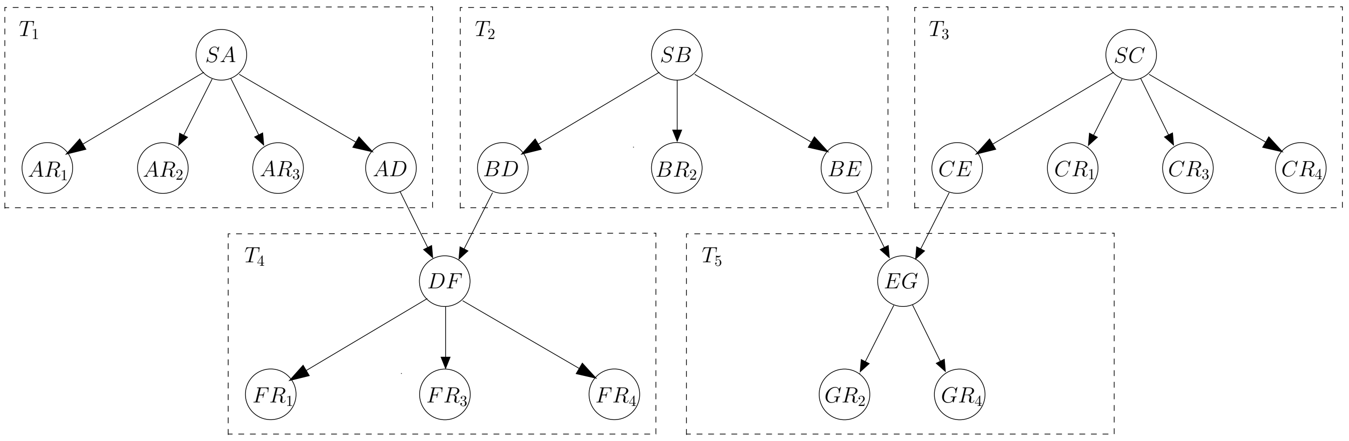


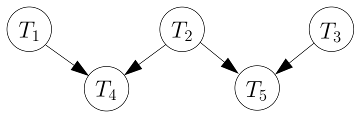
Fig. 3. The Line graph derived from Fig. 2

Consider the network in Fig. 2. The source $\mathcal{S}$ would like to multicast an ordered scalable message $\mathbf{M} = [m_1, m_2, m_3]$, of which elements m_1, m_2, and m_3 have dependency levels of 1, 2, and 3, respectively, to four sink nodes R_1, R_2, R_3, and R_4. $\mathbf{Q}(\mathcal{S}) = [1, 0.5, 0.25]$ and every edge in the graph is capable of transmitting one symbol per time unit.

Before assigning a GEK to each edge, we can simplify the graph in Fig. 2, using Fragouli, Soljanin, and Shokrollahi's approach [8]. Figure 2 is first transformed into a line graph shown in Fig. 3, where each node represents an edge from Fig. 2. Any two nodes in Fig. 3 are connected if the corresponding edges in Fig. 2 are adjacent.

The nodes in Fig. 3 are grouped into five subtrees, each of which is bounded by dashed lines, such that the members in each subtree are forced by the topology to have the same GEK. For example, in the subtree T_1, SA and AD must have the same GEK, since, according to Fig. 2, the node A has only one incoming edge SA and thus can do nothing but copy the received symbols and forward the copies to all outgoing edges AR_1, AR_2, AR_3, and AD, hence the same GEK among them.

Accordingly, our problem of assigning GEKs to twenty-one edges is reduced to that of assigning GEKs to five subtrees. The minimum subtree graph is shown in Figure 4.



$$\begin{tikzcd} T_1 \arrow[dr] & & T_2 \arrow[dl] \arrow[dr] & & T_3 \arrow[dl] \\ & T_4 & & T_5 & \end{tikzcd}$$

Fig. 4. The minimum subtree graph derived from Fig. 3

In this case, the subtrees $T_1 = \{SA, AD, AR_1, AR_2, AR_3\}$, $T_2 = \{SB, BD, BE, BR_2\}$, $T_3 = \{SC, CE, CR_1, CR_3, CR_4\}$, $T_4 = \{DF, FR_1, FR_3, FR_4\}$, $T_5 = \{EG, GR_2, GR_4\}$.

The edges connecting T_1 and T_2 to T_4, as well as T_2 and T_3 to T_5 in Fig. 4 imply that the GEK of T_4 must be derived from T_1 and T_2 whereas that of T_5 must be derived from T_2 and T_3, i.e., the sets of vectors $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_2}, \mathbf{f}_{T_4}\}$ and $\{\mathbf{f}_{T_2}, \mathbf{f}_{T_3}, \mathbf{f}_{T_5}\}$, where $\mathbf{f}_{T_x}$ denotes the GEK of the subtree T_x, must be linearly dependent.

In the same manner, linear independence constraints can be represented by the sets $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_2}, \mathbf{f}_{T_3}\}$, $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_3}, \mathbf{f}_{T_4}\}$, $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_2}, \mathbf{f}_{T_5}\}$, and $\{\mathbf{f}_{T_3}, \mathbf{f}_{T_4}, \mathbf{f}_{T_5}\}$. The three GEKs in each set must be linearly independent in order to ensure a full-rank system of linear equations at the source and each sink, when there is no erasure. For example, the sink S is connected to the subtrees T_1, T_2, and T_3, hence the linearly independence constraint $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_2}, \mathbf{f}_{T_3}\}$.

The next section deals with optimizing the scalable data quality under these constraints.

VI. A Strategy for GEK Assignment

We propose a strategy for assigning a GEK to each subtree. To achieve fairness, we use a minimax criterion, i.e., minimizing the maximum expected degradation, which is equivalent to maximizing the expected data quality of the poorest sink.

Since our example concerns allocating GEKs to five subtrees from a set of seven available GEKs, as shown in Table I, the attempt to find the optimal solution at once is impractical because it results in the search within the space of the size 7^5. Therefore, we may adopt an iterative approach, in which at the iteration t, $n(t)$ GEKs are allocated to $n(t)$ subtrees.

The assignment of $n(t)$ GEKs to $n(t)$ subtrees at the t^{th} iteration does not have the same effect on every sink. An individual sink may either 1) not be connected to any of those subtrees and thus gain nothing more than the previous iteration, or 2) be connected to certain subtrees but still cannot recover any other data than that received in the previous iteration, or 3) be connected to certain subtrees and can recover some more data. In the first and the second cases, the expected data quality of the sink does not improve at the t^{th} iteration, whereas, in the third case, it does.

In each iteration, we aim to ensure that the temporary minimax criterion is satisfied. To do so, we first have to evaluate the temporary expected data quality $E_t[Q_i]$ at this current t^{th} iteration. As earlier discussed, $E_t[Q_i]$ of each sink i can be different from others. To satisfy the temporary minimax criterion, we find the minimum $E_t[Q_i]$ among all i for every possible GEK assignment at the current iteration, and then select the assignment that gives the maximum value of min $E_t[Q_i]$.

The temporary expected data quality $E_t[Q_i]$ of the i^{th} sink at the t^{th} iteration is given by

$$E_t[Q_i] = \sum_{j=1}^{\omega} \phi_{t,i,j} \cdot \Delta_j, \tag{12}$$

where $\phi_{t,i,j}$ represents the probability that the prefix $\mathbf{P}_j$ can be recovered at the sink i after the GEK assignment at the iteration t, which equals $\rho_{i,j}$ in Eq. (8) if $\mathbf{P}_j$ can be recovered at this iteration, or equals zero if it cannot. Δ_j denotes the incremental quality if $\mathbf{P}_j$ is recovered.

We suggest that the number $n(t_1)$ of GEKs assigned at the iteration t_1 should be greater than or equal to the number $n(t_2)$ at the iteration t_2 if $t_1 < t_2$, i.e., it is better to take larger search spaces into consideration during some first iterations, after which things do not much improve.

In our network example, we assume that every edge-disjoint path has an erasure probability of 0.1 and $\Delta\mathbf{Q} = [\Delta_1, \Delta_2, \Delta_3] = [1, 0.5, 0.25]$. If we let $n(1) = 3$ and $n(2) = n(3) = 1$, then the minimum temporary expected data quality in the first iteration is maximized by allocating the GEKs $[1\ 0\ 0]^T$ to T_3, $[0\ 1\ 0]^T$ to T_1, and $[1\ 1\ 0]^T$ to T_5. The sink R_1 now has the temporary expected data quality of $E_1[Q_1] = (0.9)(1) + (0.81)(0.5) = 1.305$, since, by receiving $[1\ 0\ 0]^T$ from T_3 with the probability of 0.9, R_1 can recover the first prefix, and by receiving both $[1\ 0\ 0]^T$ from T_3 and $[0\ 1\ 0]^T$ from T_1 with the probability of 0.81, it can recover the second prefix.

In the same manner, the temporary expected data qualities $E_1[Q_2]$, $E_1[Q_3]$, and $E_1[Q_4]$ of R_2, R_3, and R_4 become 1.305, 1.305, and 1.215, respectively. This gives the minimum temporary expected data quality, min $E_1[Q_i]$, of 1.215 at the sink R_4, which is the maximum that one can find at this iteration.

Table II shows the resulting expected data quality at each node and the minimum one, after the third iteration finishes and all subtrees have obtained their GEKs.

TABLE II
THE RESULTING EXPECTED QUALITIES OF THE RECOVERED SCALABLE DATA IN THE NETWORK EXAMPLE WHEN THE SUGGESTED STRATEGY IS APPLIED

$E_3[Q_1]$	1.4873
$E_3[Q_2]$	1.4873
$E_3[Q_3]$	1.4873
$E_3[Q_4]$	1.3973
min $E_3[Q_i]$	1.3973

VII. Results and Conclusion

Table III shows that, in our network example, the suggested strategy achieves the global optimum. In addition, on average, a random GEK assignment results in the minimum expected quality that is closer to that of the worst case than the best one. Thus, a significant amount of quality can be saved if we take the UEP mechanism of network codes into consideration by applying the suggested strategy.

TABLE III
THE FOUR-CASE COMPARISON OF MINIMUM EXPECTED QUALITIES OF THE RECEIVED SCALABLE DATA IN THE NETWORK EXAMPLE

The Best Assignment	The Suggested Strategy	The Average over All Possible Assignments	The Worst Assignment
1.3973	1.3973	1.3055	1.2757

References

[1] A.E. Mohr, E.A. Riskin, R.E. Ladner, ”Unequal Loss Protection: Graceful Degradation of Image Quality over Packet Erasure Channels Through Forward Error Protection,” *IEEE J. Select. Areas Commun.*, vol. 18, pp. 819-828, Jun. 2000.

[2] R. Ahlswede, N. Cai, S.-Y.R. Li, and R.W. Yeung, ”Network Information Flow,” *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204-1216, Jul. 2000.

[3] S.-Y.R. Li, R.W. Yeung, and N. Cai, ”Linear Network Coding,” *IEEE Trans. Inform. Theory*, vol. 49, pp. 371-381, Feb. 2003.

[4] R. Koetter, and M. Mé́dard, ”An Algebraic Approach to Network Coding,” *IEEE/ACM Trans. Networking*, vol. 11, pp. 782-795, Oct. 2003.

[5] P.A. Chou, Y. Wu, and K. Jain, ”Practical Network Coding” 51st *Allerton Conf. Communication, Control, and Computing*, Oct. 2003.

[6] R.W. Yeung, S.-Y.R. Li, N. Cai, and Z. Zhang, ”Network Coding Theory” *Foundation and Trends in Communications and Information Theory*, vol. 2, nos 4 and 5, pp. 241-381, 2005.

[7] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain, L. Tolhuizen, ”Polynomial Time Algorithms for Multicast Network Code Construction,” *IEEE Trans. Inform. Theory*, vol. 51, pp. 1973-1982, Jun. 2005.

[8] C. Fragouli, E. Soljanin, A. Shokrollahi, ”Network Coding as a Coloring Problem,” in *Proc. Conf. Information Sciences and Systems.* Princeton, NJ, Mar. 2004.

[9] R.W. Yeung, and N. Cai, ”Network Error Correction, Part I: Basic Concepts and Upper Bounds” *Communications in Information and Systems*, vol. 6, no 1, pp. 19-36, 2006.

[10] N. Cai, and R.W. Yeung, ”Network Error Correction, Part II: Lower Bounds” *Communications in Information and Systems*, vol. 6, no 1, pp. 37-54, 2006.

[11] R. Koetter, and F.R. Kschischang, ”Coding for Errors and Erasures in Random Network Coding” *IEEE Trans. Inform. Theory*, 2008, to be published.