# Age of Information Analysis for Instantly Decompressible IoT Protocols

Niloofar Yazdani and Daniel E. Lucani
DIGIT and Department of Engineering, Aarhus University, Denmark
{n.yazdani, daniel.lucani}@eng.au.dk

***Abstract*—Generalized deduplication (GD) has been proposed as a new approach for reducing the cost of storage. Recent work has adapted this technique to provide distributed, multi-source lossless compression to reduce the total number of bits transmitted in sensor networks. In this paper, we characterize its performance and advantages from an age of information perspective. For simplicity, we analyze the case of one source node receiving one symbol/sample per unit time and transmitting bits to the sink node. We show the potential for GD to also deliver instant decoding of the data to further reduce the average age of information. Using real-world data sets, our solution reduces the information age by 25% and 36% when considering the standard and the instantly decodable versions, respectively compared to the use of the DEFLATE algorithm for compression.**

***Index Terms*—timeliness, age of information (AoI), wireless sensors, Generalized Data Deduplication, data transmission reduction.**

## I. Introduction

A growing number of applications from live video streaming to haptic applications and Industry 4.0 impose stringent requirements to deliver data from the source to the destination in a timely manner. The concept of information age [1] was proposed to quantify the freshness of information at the sink node. The age of information (AoI) is an indication of the time interval between the present time and when the most up-to-date symbol at the sink node is generated. Various metrics have been considered to characterize the information age for different system models. For example, Zhong et al. [2] studied the information age for a lossless source coding system where symbols arrive randomly, particularly, studying the average delay of the proposed fixed-to-variable block coding scheme. Inoue et al. derived a general formula for the stationary distribution of the age of information which can help for the analysis of complicated systems [3]. Recent work extended the age analysis to a wireless powered sensor network, where an update is generated when the battery is fully charged [4].

In order to reduce the AoI, recent schemes have considered online compression strategies for the data source using common dictionaries in the sink and source nodes. These assume prior knowledge of the data source to deliver good performance. More advanced compression techniques require collecting and analysing enough data to provide sufficient compression potential [2]. The AoI might be affected negatively by the required time to have this amount of data, the time to compress it, and later decompress it. Thus, there is a need for compression techniques that (i) are oblivious to the data source statistics, and (ii) provide good compression potential for small data blocks in order to reduce the information age.

This work was supported in part by the SCALE-IoT Project (Grant No. 7026-00042B) granted by the Danish Council for Independent Research, the Aarhus Universitets Forskningsfond Starting Grant Project AUFF- 2017-FLS-7-1.

Generalized deduplication (GD) was recently proposed as an alternative to data deduplication [5], [6] as well as for other lossless compression schemes to efficiently compress data [7]. In [8], we proposed a new lossless, multi-source data compression approach to reduce the amount of data transmission that uses an indexing technique inspired by GD. In fact, the technique allows for fast decompressing of IoT data samples. In this paper, we focus on characterising the average of information age using the proposed transmission protocol in [8] and adaptations of it that help reduce the AoI, e.g., with or without memory at the sensor node.

More precisely, we propose an *instantly decodable* (IDe) variation that reorganizes the data to decode every internet of things (IoT) sample as it arrives as part of an encoded block of data (a compressed block containing several samples). The approach is simple, does not increase the complexity at time of compression or decompression, and reduces the average of information age significantly. Our analysis also provides basic upper and lower bounds for the average of information age. Finally, we provide numerical results on mathematical analysis based on the model from [8] and also for a real-world data set showing significant gains in compression gain and average of information age with respect to state-of-the-art technique.

## II. Generalized Deduplication for Efficient Data Transmission

GD uses a transformation function to take each data symbol or data block into a basis and a deviation. In our context, this transformation process takes place at the source node, e.g., a sensor node, with the intention to send the basis (large) only if not available at the sink node (e.g., Edge device, Cloud) to reduce the amount of transmitted data, while the deviation is always transmitted.

More specifically, each source node generates a series of data blocks, $C_i$'s, to be sent to the sink node. By applying GD, each $C_i$ is transformed into a pair $(b_i, d_i)$. Rather than sending the content of $C_i$, the bases, $b_i$, and the information regarding the associated deviations, $d_i$, can be sent. To achieve efficient data transmission, the source node will first send a basis ID, e.g., computed using a hash function that is common to all sources, and the compressed deviation for each piece of data ($C_i$). If sink node has the basis for the transmitted basis ID, it saves the new information. Otherwise, it requests for the basis, itself. Each source node benefits from all bases IDs at the sink node, whether it is generated by that source node or not.

For creating the mapping from $C_i$ (input) to $(b_i, d_i)$ an error correcting code (ECC) can be used. $C_i$'s are considered as the codeword and by applying the decoding algorithm of the ECC, the message word recovered is the basis. The difference between the codeword and the error-free codeword, e.g., encoding basis $b_i$ using the ECC, is the deviation. In [8], we used a Hamming decoder for creating the mapping from $C_i$ (input) to the pair $(b_i, d_i)$. With $m$ being the number of parity bits, $n = 2^m - 1$ is the bit-length of the data block in our context, and $k = 2^m - m - 1$ is the length of the basis. We also define data block length in bytes as $n_B = \lceil \frac{n}{8} \rceil$. For representing the deviation, $m$ bits are sufficient to convey the location of the bit error as a Hamming code corrects only one bit error. In our context, this means that all data blocks mapped to a given basis are at most 1 bit away from the error-free data block. In order to generate compact basis IDs, various algorithms can be considered, e.g., **SHA-1** and **CRC32** to generate basis IDs of length 160 or 32 bits, respectively.

In this paper, we study the age of information for the lossless data transmission compression method based on GD originally proposed in [8] and variations of it. We use a Hamming ECC as the transformation function. We study two different cases: a) a source node that has enough memory to save all the basis IDs it has generated; and b) a memory-less source node, as originally envisioned in [8]. In (b), the source node sends the basis IDs first to check if they are available at the sink node. If a basis ID is not recognized, the basis is transmitted by the sensor node. In case (a), the source node only sends a basis ID if it has already been sent to the sink node by that sensor, which helps in reducing the time to communicate the data. For our system, it is clear that the input is of fixed size, $n$ bits, while the output has two possible lengths if there is a match to previously sent data to the sink node, $L_1$, or otherwise, $L_2$. For the former, the basis does not need to be sent, that is

$$L_1 = s + d + h, \tag{1}$$

where $h$ and $s$ are the number of bits required to represent the length of the basis ID and signalling bits, respectively.

To compute $L_2$, we consider two cases. If source node has enough memory, it is aware of the situation and does not send the basis ID ($h$ bits). Thus,

$$L_2 = s + k + d + [h], \tag{2}$$

where $[h] = h$ is for a memory-less source node and $[h] = 0$ otherwise. For a memory-less source node, $s = 1$ bit. When the source node is idle, it transmits a single bit `0`. Otherwise, it sends a `1` followed by the encoded message. For a source node with memory, $s = 2$ bits. The extra bit is to determine the type of the encoded block (with a basis ID or basis). Classical deduplication ($DD$) is a special case of $GD$, where there is no deviation, i.e., $d = 0$, $m = 0$, $k = n$.

## III. Age of Information Analysis

We study the timeliness of our scheme following a similar approach to the streaming source coded analysis in [2] of streaming source coding. A single source node is considered as a worst case scenario for our approach. Additional source nodes have the potential to increase the knowledge at the sink node and result in reductions of the age of information on other sources. We assume a constant rate of $R$ between the encoder (the source node) and the decoder (the sink node) without propagation delay. One source symbol is generated per unit time. The encoder maps every $B$ source symbols into variable length bit strings. The delivery time of the $j-th$ encoded block $Y_j$ is indicated by $D_j$ as in Fig. 1. If the decoder's most recently decoded symbol at time $t$ is generated at time $i$, the instantaneous status age is $\Delta(t) = t - i$. Then, the time-average status age, $\Delta$, is

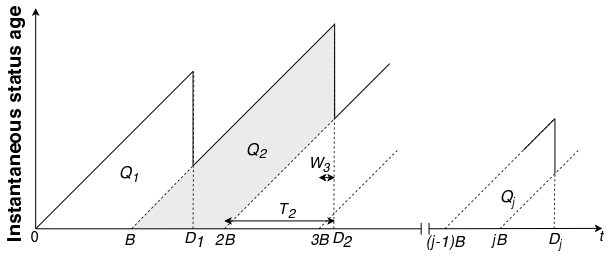


Fig. 1. Example of status age of streaming fixed-to-variable length coding.

$$\Delta = \lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \int_0^{\mathcal{T}} \Delta(t) d(t). \tag{3}$$

To calculate the integral, we follow the same approach as in [9] to calculate the sum of trapezoids areas $Q_j$, depicted in Fig. 1. Assuming $N$ as the number of encoded blocks,

$$\Delta = \lim_{N\to\infty} \frac{1}{BN} \sum_{j=1}^{N} Q_j = \lim_{N\to\infty} \frac{1}{N} \sum_{j=1}^{N} (D_j - jB) + \frac{B}{2}, \tag{4}$$

where $D_j - jB$ is the time interval between the $j-th$ encoded block is ready at the encoder and it is received by the decoder, which is equal to the addition of waiting time ($W_j$) and service time ($S_j$) for that block, i.e., $D_j - jB = W_j + S_j$. The *waiting time* is related to the time that the encoder's FIFO buffer is busy transmitting other encoded blocks while the *service time* is the required time for an encoded block of length $L_j$ to be transmitted, $S_j = \frac{L_j}{R}$.

The waiting time $W_j$ for GD, comes from two sources. First, the encoder's FIFO buffer is busy with transmitting other encoded blocks, $W_{b,j}$. If the source node finishes transmitting each encoded block before arrival of the next one, then $W_{b,j} = 0$. Second, the source node is waiting for the sink node basis request because the basis ID is not available at the sink node, $W_{r,j}$. Note that $W_{r,j} = 0$ if a source node with memory or for a memory-less source node where basis ID is available at the sink node. Otherwise, $W_{r,j}$ is mainly dependent to the length of the request message and $R$, i.e., $W_{r,j} = W_r$, where $W_r$ is a constant in our system model.

We study how the average of status age changes as time passes. Using the approach in [9], the time average age over an interval $(0, \mathcal{T})$ is defined as

$$\Delta_{\mathcal{T}} = \frac{1}{\mathcal{T}} \int_0^{\mathcal{T}} \Delta(t) d(t), \tag{5}$$

where the area defined by the integral is the concatenation of the trapezoids $Q_j$ and the last triangular area of width $T_N$.

Following the same approach as in [2], [9] and assuming the use of streaming source coding and lossless fixed-to-variable block coding, we define the average of status age at the time of receiving the $N-th$ encoded block by the decoder as

$$\begin{aligned}\Delta_{ave}(N) &= \frac{1}{BN}\int_0^{BN}\Delta(t)d(t) = \frac{1}{BN}[\sum_{j=1}^{N}Q_j + \frac{T_N^2}{2}] \\ &= \frac{1}{N}\sum_{j=1}^{N}(D_j - jB) + \frac{B}{2} + \frac{(D_N - NB)^2}{2BN},\end{aligned} \tag{6}$$

where the third term in Eq. 6 will be negligible as $N$ grows to a stable queue due to the finite value of $D_N - NB$.

For the specific case of GD, $D_j - jB$ in Eq. 6 is equal to:

$$D_j - jB = \begin{cases}\frac{L_1}{R} & match \\ \frac{L_2}{R} + [W_r] & no\ match\end{cases} \tag{7}$$

### A. Timely Throughput Condition

In this paper, we choose $R$ value such that the longest encoded block is sent before arrival of the next one ($B$ time units):

$$\frac{L_{max}}{R} + [W_r] < B \Rightarrow R > \frac{L_{max}}{B - [W_r]} \Rightarrow R > \frac{L2}{B - [W_r]}. \tag{8}$$

This condition is stricter than a condition on stable throughput. The rationale behind this choice is that we only have information about the worst case transmission rate at the start of the process. Given a sink node that stores all previously transmitted data, the rate $R$ could be adapted to more accurately match the system's state. This adaptive $R$ selection will be studied in our future work.

### B. instantly decodable strategy

In [2], the assumption is that all the source symbols in an encoded block are decoded together, once all the encoded block is received by the sink node. The idea is to organize the data in an encoded block to transmit it in a way that the sink node is able to start decoding the data sooner for the case that bases are not available at the sink node. For example, if $C_i$ contains four source symbols $X_1$, $X_2$, $X_3$, and $X_4$ as shown in Fig. 2 (a) and we use a Hamming ECC as a transformation function ( see Sec. II). If the source node transmits the deviation bits first and then the bases from the left, as shown in Fig. 2 (b), the decoder is able to decode the data symbol by symbol as it receives the bit strings as depicted in Fig. 2 (c) by purple solid lines. That is because the sink node will be aware of which bit to flip after receiving the deviation. This comes from the fact that the deviation of a Hamming ECC already contains the information related to the position of the bit to be flipped.

In Fig. 2. (c), the encoder groups every $B$ symbols. After receiving $B$ symbols, a new encoded block is ready for transmission. Using our new structure, we do not decode all the source symbols of an encoded block, $Y_j$, at time $D_j$, as in previous work. Instead, the symbols will be decoded one by one with only a minor delay introduced at the time of transmitting the deviation information at the beginning of the process.

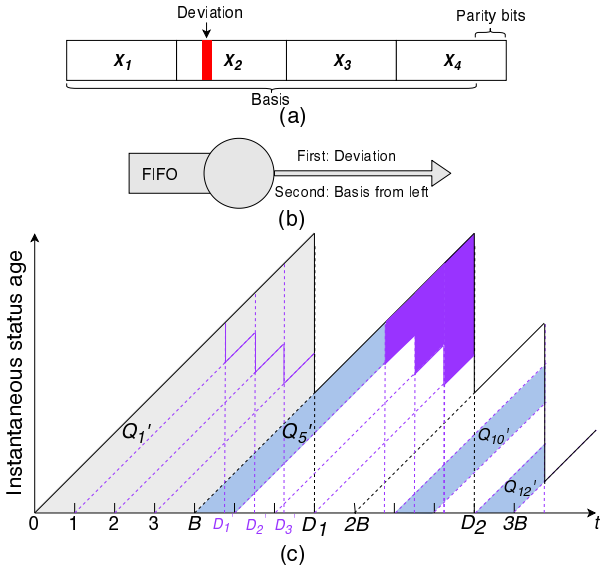


Fig. 2. Example of the proposed mechanism to reduce the information age, IDe, for $B = 4$ and for using a Hamming ECC as a transformation function.

$D'_i$ is the decoding time of the $i-th$ source symbol. To calculate the $\Delta_{ave}(N)$, we need to calculate the blue areas, $Q'_i$, instead of the grey area. The purple area in Fig. 2 (c) demonstrates the reduction in average information age by applying our *instantly decodable* strategy for GD.

Note that if the basis linked to the basis ID is already available at the sink node, similar to the third transmission in Fig. 2 (c), decoding time of all the symbols of the encoded block are the same (endpoint of $Q'_{10}$ and $Q'_{12}$ are equal).

Following the same approach as in [9] to calculate the sum of trapezoids areas $Q'_i$ and considering $N' = BN$, $B' = 1$:

$$\begin{aligned}\Delta_{ave}(N) &= \frac{1}{N'}[\sum_{i=1}^{N'}Q'_i + \frac{(D'_N - N')^2}{2}] \\ &= \frac{1}{N'}\sum_{i=1}^{N'}(D'_i - i) + \frac{1}{2} + \frac{(D'_{N'} - N')^2}{2N'}\end{aligned} \tag{9}$$

Let us define $M = \frac{n}{B.R}$ as the time interval between arriving of symbols of an encoded block at the decoder and $\mathcal{X}_i = i\ mode\ B$. Then,

$$D'_i - i = \begin{cases}\frac{L_1}{R}\ match,\ \mathcal{X}_i = 0 \\ (B - \mathcal{X}_i) + \frac{L_1}{R}\ match,\ \mathcal{X}_i \neq 0 \\ \frac{L_2}{R} + [W_r]\ no\ match,\ \mathcal{X}_i = 0 \\ (B - \mathcal{X}_i) + \frac{L_2 - k}{R} + \mathcal{X}_i.M + [W_r]\ no\ match,\ \mathcal{X}_i \neq 0\end{cases} \tag{10}$$

### C. Performance Bounds

The performance of $\Delta_{ave}(N)$ has clear upper and lower bounds given the structure of our problem. A trivial lower

bound can be found by considering that every basis ID matches to an already existing basis ID, i.e., no basis transmission is required. This happens when number of blocks is large enough. For this case, the length of an encoded block is always equal to $L_1$. Based on Eq. 6 and Eq. 7 and considering the third item in Eq. 6 is temporary for a stable buffer,

$$\Delta_{ave}(N) \geq \Delta_{min} = \frac{L_1}{R} + \frac{B}{2}. \tag{11}$$

An upper bound can also be found by considering the case of having no repeated data, i.e., every single basis must be transmitted to the sink node, then

$$\Delta_{ave}(N) \leq \Delta_{peak} = \frac{L_2}{R} + [W_r] + \frac{B}{2} + \frac{(\frac{L_2}{R} + [W_r])^2}{2B}, \tag{12}$$

where $[W_r]$ is non-zero only for a memory less source node and we evaluate the last term in Eq. 6 at $N = 1$.

If we consider a large enough $N$, then

$$\lim_{N\to\infty} \Delta_{ave}(N) \leq \Delta_{max} = \frac{L_2}{R} + [W_r] + \frac{B}{2}. \tag{13}$$

For the instantly decodable scheme, the upper bound can be refined. Using a similar strategy as for the standard upper bound, this means $D'_i - i$ is periodic (not constant) over the period of $B$. Using Eq. 9 in steady state where the last term is gone:

$$\Delta_{max,IDe} = \frac{1}{B}\sum_{i=1}^{B}(D'_i - i) + \frac{1}{2}, \tag{14}$$

where,

$$D'_i - i = \begin{cases} \frac{L_2}{R} + [W_r] & for\ i = B \\ (B-i) + \frac{L_2 - k}{R} + i.M + [W_r] & for\ i < B, \end{cases} \tag{15}$$

according to the Eq. 10 for the case of having no match.

*Example 1:* If $B = 4$, $m = 7$, $R = 42$ [bits]/[Unit Time], the upper bound for a source node with memory ($W_r = 0$) is 5.04 and 4.04 for the standard GD and the instantly decodable GD, respectively. This means 19.78% reduction in the upper bound.

## IV. Numerical results

### A. Comparison Schemes

We are interested in comparing the age of information for our data transmission compression method using various schemes for transformation and indexing of the data as well as state-of-the-art techniques for compression.

**DEFLATE:** DEFLATE [10] is a well-known compression technique which is a combination of LZ77 [11] and Huffman [12].

**Differential DEFLATE:** Differential DEFLATE (DEFLATE_diff) applies DEFLATE to the difference between each sample and the previous one except for the first sample which is kept unchanged before compression is applied.

**GD:** GD maps each data block to a basis and a deviation using a Hamming ECC as a transformation function. The basis is transmitted only if it is not available at the sink node to reduce the amount of data transmission. Otherwise, the basis ID corresponding to the basis and the deviation are transmitted to the sink node.

**DD:** DD is a special case of GD where there is no deviation, i.e., the whole block is considered as a basis. The block is transmitted only if not available at the sink node.

**AGD:** This scheme introduced in [8] learns basic parameters from the data and performs a shift to align the data mean of the data with a error-free data block prior to applying GD. This technique increases the potential for compression by activating fewer basis in the system. The information regarding the shift is sent as $m$ additional bits in the transmission (aside from the standard $m$ bits for the deviation).

**Differential GD, differential DD:** Differential GD (GD_diff) and differential DD (DD_diff) apply GD and DD, respectively, to the difference between each sample and the previous one except for the first sample which is kept unchanged before compression.

**IDe:** By IDe, we identify cases where we have used our instantly decodable scheme.

### B. Performance Bounds

Fig. 3 shows the performance bounds with respect to $R$ for $B = 16$, $h = 32$, $n_B = 32$ $B$, and a length of request message of 24 bits so that $W_r = \frac{24}{R}$. Fig. 3 considers cases both memory-less sources and sources with memory as well as the effect of instantly decodable scheme. $\Delta_{min}$ is similar for both cases. However, other metrics increase by $W_r + \frac{h}{R}$ for the memory-less case. While $DD$ (classical deduplication) is slightly better in terms of status age, we will show that it normally takes a longer time for it to reach the better status age compared to other schemes. Finally, introducing instantly decodable strategy reduces the upper bound on $\Delta$ by up to 28%.

### C. Analytical Data Model

We consider the data model and probability analysis in [8]. Particularly, a data model considers that the $i$-th sample/data block is $C_i = A + e_i$, where $A$ is constant and each bit can be flipped with probability $q$ ($0 \leq q \leq 1$) and independent of other bits, and the addition is carried out as a bit-by-bit XOR. Each $e_i$ vector contains $e$ non-zero bits. This means that the probability of having a $C_i$ with $e$ bit flips with respect to $A$ and length $n$ bits is

$$P(e) = q^e \cdot (1-q)^{(n-e)} \cdot \binom{n}{e}, \text{for } e = 0, 1, 2..., \tag{16}$$

and zero otherwise. As described in [8], the *matching probability* for the $N$-th received data block is the probability of finding a match to the $N-1$ previously stored data blocks at the sink node. The matching probability for the $N$-th data block with $e$ bit flips is

$$f_N(\gamma(e)) = 1 - (1 - \gamma(e))^{N-1}, \tag{17}$$

where $\gamma(e)$ is the probability of a data block with $e$ bit flips to match another data block given the policy, e.g., GD, AGD, or DD.

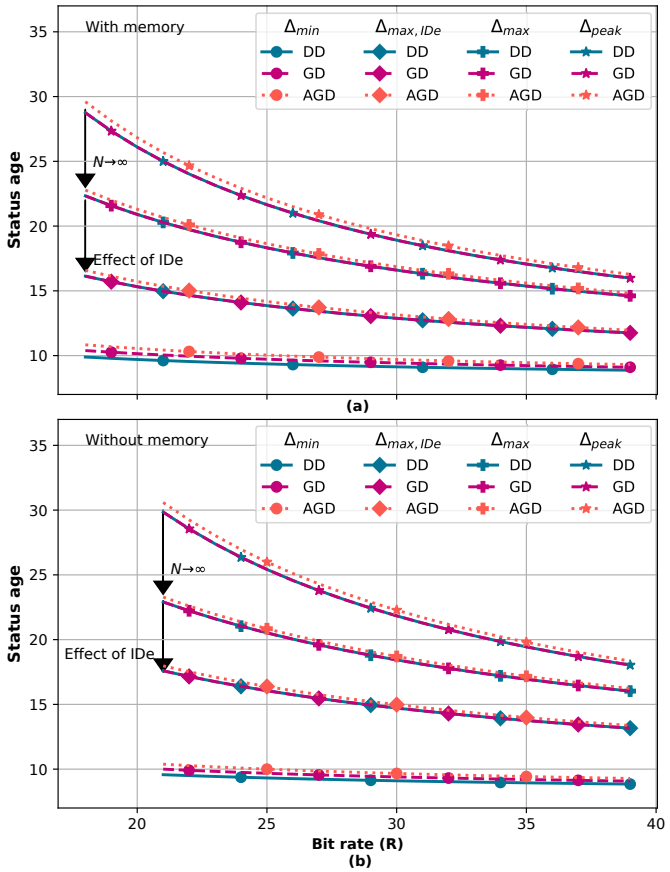


Fig. 3. Boundaries for $B = 16$, $h = 32$, $W_r = 24/R$, and packet length of 32 B for: a) source node has memory, b) source node is memory-less.

**Matching Probability of Deduplication:** The probability of the $N$-th data block matching a previous data block is

$$\mathbb{P}_{DD}(N) = \sum_{e=0}^{n} P(e) \cdot f_N\left(\frac{P(e)}{\binom{n}{e}}\right). \tag{18}$$

**Matching Probability of GD:** The functions $\gamma_1(e)$, $\gamma_2(e)$ and $\gamma_3(e)$ are the probability of a data block with $e$ bit flips to match another data block for the three different potential cases [8] for GD

$$\begin{aligned}
\gamma_1(e) =& P(e-1) \cdot \frac{\binom{e}{e-1}}{\binom{n}{e-1}} + \frac{P(e)}{\binom{n}{e}} + P(e+1) \cdot \frac{\binom{e+1}{e}}{\binom{n}{e}} \\
\gamma_2(e) =& P(e-2) \cdot \frac{\binom{e-1}{e-2}}{\binom{n}{e-2}} + \frac{P(e-1)}{\binom{n}{e-1}} + P(e) \cdot \frac{1}{\binom{n}{e}} \\
\gamma_3(e) =& P(e) \cdot \frac{\binom{e+1}{e}}{\binom{n}{e}} + \frac{P(e+1)}{\binom{n}{e+1}} + P(e+2) \cdot \frac{\binom{e+2}{e+1}}{\binom{n}{e+1}},
\end{aligned} \tag{19}$$

Accordingly, the matching probability of the $N$-th data block is

$$\begin{aligned}
\mathbb{P}_{GD}(N) = \sum_{e=0}^{n} P(e) \cdot \Big( & f_N(\gamma_1(e)) 2^{-m} + (2^m - 1) 2^{-2m} \cdot \\
& f_N(\gamma_2(e)) + (1 - 2^{-m})^2 \cdot f_N(\gamma_3(e)) \Big).
\end{aligned} \tag{20}$$

**Matching probability of AGD:** Matching probability of the $N$-th data block is

$$\begin{aligned}
\mathbb{P}_{AGD}(N) = & \sum_{e=0}^{1} P(e) \cdot f_N(P(0) + P(1)) + P(2) \cdot f_N(\gamma_3(2)) \\
& + P(3) \cdot 2^{-m} \cdot (f_N(\gamma_1(3)) + (2^m - 1) \cdot f_N(\gamma_3(3))) + \sum_{e=4}^{n} \frac{P(e)}{2^{2m}} \cdot \\
& \Big( f_N(\gamma_1(e)) \cdot 2^m + (2^m - 1) \cdot f_N(\gamma_2(e)) + (2^m - 1)^2 \cdot f_N(\gamma_3(e)) \Big).
\end{aligned} \tag{21}$$

$M_{\mathcal{P}}(N)$ is the total number of matches for policy $\mathcal{P}$ such that

$$M_{\mathcal{P}}(N) = \sum_{i=1}^{N} \mathbb{P}_{\mathcal{P}}(i). \tag{22}$$

Accordingly, $\Delta_{ave}(N)$ is calculated by using Eq. 6 and 7, where

$$\sum_{j=1}^{N} (D_j - jB) = M_{\mathcal{P}}(N).\frac{L1}{R} + (N - M_{\mathcal{P}}(N)).(\frac{L_2}{R} + [W_r]). \tag{23}$$

Considering the worst case for the last term, i.e., $j = N$, then

$$D_N - NB = \frac{L_2}{R} + [W_r]. \tag{24}$$

The parameter $[W_r]$ is non-zero only for a memory-less source node and we assume it equal to $\frac{208}{R}$ for a basis request message of 26 B (208 bits). Fig. 4 shows the $\Delta_{ave}(N)$ for our data model for block length of 256 B, $B = 1$, $h = 32$, $R = 2350$ and $q = 0.00025$. $\Delta_{ave}(N)$ for a memory-less source node is higher at the beginning due to the $W_r$ and extra hash bits ($\frac{h}{R}$) transmission. As the probability of matching increases the $\Delta_{ave}(N)$ decreases. With only 10 blocks, we have around 44% and 59% reduction in $\Delta_{ave}(N)$ for DD/GD and AGD, respectively.

### *D. Real-world data set*

Fig. 5 compares the compression gain and $\Delta_{ave}(N)$ of our technique under differential GD and differential DD schemes with differential DEFLATE considering real-world data set obtained by the Intel Berkeley Research Laboratory [13]. We considered temperature data of all sensor nodes with accuracy of one digit after the decimal point and for a temperature range of 15.0 to 32.0. Samples are of 8 bits, $B = 16$, and $R = 10$. We assumed a source node with enough memory to keep all the bases IDs.

The compression gain for differential DEFLATE is almost constant which means it has nearly constant $\Delta_{ave}(N)$ versus $N$. The beginning variation in $\Delta_{ave}(N)$ in all schemes is due to the temporary effect of the last item in Eq. 6, which vanishes by having almost 10 blocks. It takes about 60 blocks for our scheme to find matches to provide a good gain. Thus, for $N < 90$, differential DEFLATE has a better $\Delta_{ave}(N)$ compared to differential GD and differential DD. However, by applying our IDe strategy, we outperform all other strategies in terms of $\Delta_{ave}(N)$. In fact, accounting for IDe decreased the $\Delta_{ave}(N)$ by up to 25%. Finally, our schemes provide up to 25% and

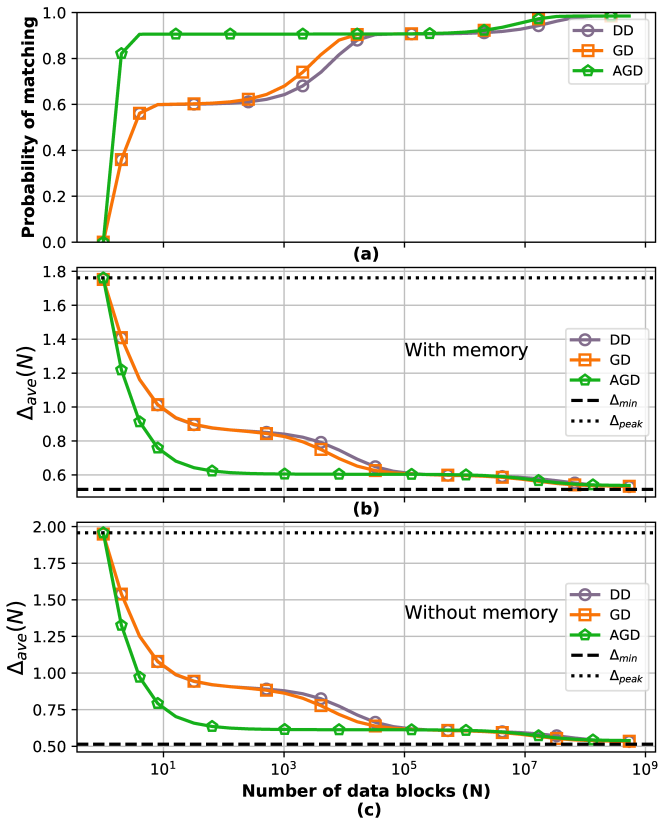


Fig. 4. $\Delta_{ave}(N)$ for $n_B$ = 256 B, $B$ = 1, $h$ = 32, $W_r = \frac{208}{R}$, $R$ = 2350 and $q$ = 0.00025 for: b) source node has memory, c) source node does not have memory.

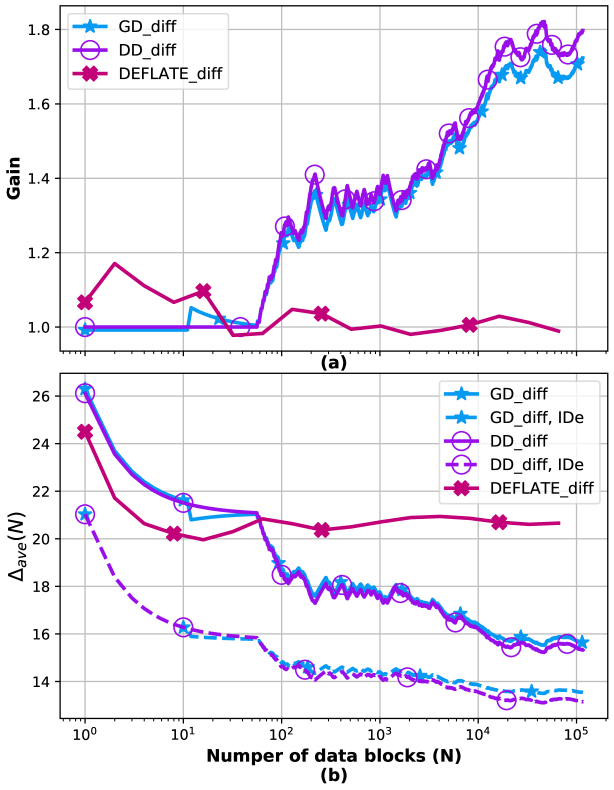


Fig. 5. Compression gain and $\Delta_{ave}(N)$ for differential compression schemes considering real-world data set. The source node has memory, $n_B$ = 16 and $R$ = 10. 8 bits samples have 1 digit after decimal point and are between 15.0 and 32.0.

36% better information age over DEFLATE with and without IDe, respectively.

## V. Conclusions

We analyzed the average of information age over time for the lossless compression protocol based on GD. We also proposed an efficient and simple scheme to provide instant decoding of the data, thus, reducing the average age of information further by 25% without introducing additional complexity in the system. Our technique learns from the data previously transmitted to rapidly reduce the age of information of future samples being measured. In fact, our analysis and numerical results show that the average age of information is reduced even when only a few samples have been previously transmitted, e.g., up to 59% for 10 previous samples. Our future work will consider alternative transformation functions for GD as well as studying the effect of memory limitations and pre-set dictionaries at the source and sink nodes that could further reduce the average age of information by limiting the need to transmit common basis in those dictionaries.

## References


[1] D. S. Alberts and D. S. Papp, "The information age: An anthology on its impact and consequences," Office of the Assistant Secretary of Defense Washington DC Command and ..., Tech. Rep., 1997.

[2] J. Zhong and R. D. Yates, "Timeliness in lossless block coding," in *2016 Data Compression Conference (DCC)*. IEEE, 2016, pp. 339–348.

[3] Y. Inoue, H. Masuyama, T. Takine, and T. Tanaka, "A general formula for the stationary distribution of the age of information and its application to single-server queues," *IEEE Transactions on Information Theory*, 2019.

[4] I. Krikidis, "Average age of information in wireless powered sensor networks," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 628–631, 2019.

[5] R. Vestergaard, Q. Zhang, and D. E. Lucani, "Generalized Deduplication: Bounds, Convergence, and Asymptotic Properties," in *IEEE GLOBECOM (accepted, to appear)*, Waikoloa, USA, dec 2019. Available at arXiv:1901.02720 [cs.IT].

[6] L. Nielsen, R. Vestergaard, N. Yazdani, S. Talasila, D. Lucani Rötter, and M. Sipos, "Alexandria: A proof-of-concept implementation and evaluation of generalised data deduplication," *Globecom. I E E E Conference and Exhibition*, Dec. 2019, (accepted, to appear).

[7] R. Vestergaard, D. E. Lucani, and Q. Zhang, "Generalized deduplication: Lossless compression for large amounts of small IoT data," in *European Wireless Conference*. IEEE, 2019.

[8] N. Yazdani and D. E. Lucani, "Protocols to Reduce CPS Sensor Traffic using Smart Indexing and Edge Computing Support," in *IEEE GLOBECOM 2019 Workshop on Edge Computing for Cyber Physical Systems*, Dec. 2019, (accepted, to appear).

[9] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 2731–2735.

[10] P. Deutsch, "DEFLATE compressed data format specification version 1.3," Tech. Rep., 1996.

[11] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, May 1977.

[12] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[13] Intel Lab Data. [Accessed: 2019-06-15]. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html