On the Network-Wide Gain of Memory-Assisted Source Coding

Mohsen Sardari, Ahmad Beirami, Faramarz Fekri
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332
Email:{mohsen.sardari, beirami, fekri}@ece.gatech.edu

***Abstract*—Several studies have identified a significant amount of redundancy in the network traffic. For example, it is demonstrated that there is a great amount of redundancy within the content of a server over time. This redundancy can be leveraged to reduce the network flow by the deployment of memory units in the network. The question that arises is whether or not the deployment of memory can result in a fundamental improvement in the performance of the network. In this paper, we answer this question affirmatively by first establishing the fundamental gains of memory-assisted source compression and then applying the technique to a network. Specifically, we investigate the gain of memory-assisted compression in random network graphs consisted of a single source and several randomly selected memory units. We find a threshold value for the number of memories deployed in a random graph and show that if the number of memories exceeds the threshold we observe network-wide reduction in the traffic.**

I. Introduction

Several studies have demonstrated the existence of considerable amount of redundancy in the Internet data traffic, where a few major dimensions have been identified as the main sources of redundancy in the network traffic. For example, the contents of a web server contains more than 60% redundant data on the average within a ten-day period [1]. Further, there are some popular files in each server that may be requested by several clients in the network. This redundancy in the data may be leveraged to reduce the communication within the network. The existing redundancy elimination techniques are mostly based on end-to-end caching mechanisms, where the redundant content is only cached in the server and the client for future reference [2]. However, these end-to-end approaches do not efficiently leverage the redundancy in the network because there is no memorization in today's Internet except end-to-end caching mechanisms [1].

Recently, a few studies considered the deployment of redundancy elimination techniques within the network [3], [4], where the intermediate nodes in the network have been assumed to be capable of caching of the previous communication and processing of the data. These works studied the network flow reduction via ad-hoc solutions such as deduplication of the repeated segments of the traffic without any connection to the information theory. The objective of this paper is to study this problem from an information theoretic point of view. We assume that some intermediate nodes (referred to as memory nodes) are capable of the memorization (to be defined later) of the previous communications which have passed through them. We further assume that the memorized content will be used in a *memory-assisted source coding* in the network, which we refer to as *network flow compression with memory*. However, several questions remain open regarding memory-assisted compression of the network flow. Does the deployment of memory in the network provide any fundamental benefit over end-to-end solutions? How much saving could be achieved using network compression? How can the savings be achieved?

This paper attempts to answer the above fundamental questions. To the best of our knowledge, this is the first work which addresses the memory-assisted redundancy elimination of the flow from the information theoretic point of view. The data that is transmitted inside the network have different spatial and temporal probability distributions. Thus, prior knowledge of the probability distributions underlying the contents may not be assumed. Hence, one important characteristic of any compression solution is that it must be *universal* in the sense that it must be able to remove redundancy without knowing the statistics and nature of the data [5], [6].

In this paper, we focus our study on the fairly broad class of parametric information sources, which include the class of Markov sources of any finite order [7]. We formulate the problem as memory-assisted compression of the network flow, where the redundancy in the traffic data is to be removed. In this context, the redundancy elimination could be viewed as universal source coding, where the goal is to represent the data with a minimum length codeword [7]. However, as we will discuss in Sec. II, this problem is different from the distributed source coding problem because of memory units. We investigate the fundamental gain of the memory-assisted network flow compression over end-to-end universal compression techniques, where the content is compressed at the server and routed via routers without any memorization inside the network. Throughout this paper, we focus on the problem involving a *single source* (content server) that is fixed in the network and extensions to multiple sources is left as future work.

In what follows, we first describe the memory-assisted source coding in Sec. II. The memory deployment problem and the gain of memorization in networks is discussed in Sec. III. In Sec. IV we study memory-assisted network flow compression on Erdős-Rényi (ER) random graphs and find a threshold value for the number of memory units. Finally, simulation results are provided in Sec. V.

II. Memory-Assisted Source Coding

In what follows, we introduce the memory-assisted source coding via the sample network depicted in Fig 1, which is consisted of a server S, a memory unit μ (which also acts as a router), and the clients C_1 and C_2. We assume that the communication is as follows. First, client C_1, who has not previously communicated with the server, acquires the sequence f_1 from the server through the intermediate node μ. Next, client C_2, who also has not previously communicated with S, acquires the sequence f_2 from S through μ. In order to show the benefits of the deployment of memory in the router, we compare three schemes:

- *NcompNmem* (No compression with no memory), which does not apply any compression and does not utilize the memory unit.
- *UcompNmem* (Universal compression with no memory), which only applies end-to-end universal source coding at the source without using the memory unit.
- *UcompWmem* (Universal compression with memory), which assumes that the router has memory and utilizes the memory unit when compressing the data at the source.

In all of the above scenarios, we assume that the client has no previous communication with the server since it is usually the case in networks. On the other hand, the memory/router is capable of memorizing the communication of f_1 between S and C_1 in order to better compress f_2 (on the link from S to μ) that is then being delivered to C_2. We will later demonstrate that even if sequences f_1 and f_2 are *independent* given that the source model is *known*, the memorization of f_1 in μ can result in the reduction of the communication for the transfer of f_2 from S to C_2. This seemingly counter intuitive phenomenon is due to the fact that the source model is *not* known a priori (at μ). The underlying source coding must be universal, which imposes a compression overhead when the length of the sequence is finite (small) [7]. On the other hand, sequence f_1 does indeed contain some information about the unknown source parameters to the extent that an infinite length sequence f_1 can be used to identify all of the unknown parameters of the source. This side information can be memorized at the memory unit μ and the source S for the compression of f_2. Then the memory unit can decode f_2 using the side information and send f_2 to C_2. It is important to note that the saving of memory-assisted compression in terms of flow reduction is observed in the $S-\mu$ link. For example, if f_1 and f_2 are unit size and the memorization helps to compress f_2 by a factor of 2, the total flow is reduced from $1+1$, to $0.5+1$ bit$\times$hop, where there is a gain 2 in the link between S and μ.

While relevant, the memory-assisted source coding problem is different from those addressed by distributed source compression techniques (i.e., the Slepian Wolf problem) that target multiple correlated sources sending information to the same destination [8], [9]. As described in the above example, the memory-assisted source coding gain is due to the fact that the source parameter is unknown. Therefore, when the length of the sequence f_2 increases, the memory-assisted source coding gain, with respect to UcomNmem, vanishes since the source parameter can be well estimated using the sequence, and hence, f_2 can be compressed to fundamental limit (i.e., entropy rate). On the other hand, in the Slepian-Wolf, the gains are achievable in the asymptotic regime. Further, the memorization of a sequence f_1 that is independent of the sequence f_2 can result in a gain in memory-assisted compression of f_2 whereas, in the Slepian-Wolf problem, the gain is due to the bit by bit correlation between the two sequences. The memory-assisted compression problem is also distinct from the lossless source coding techniques such as LZ and CTW that simply remove redundancy in a single piece of content without regard to any memorization at the intermediate node μ [5], [6].

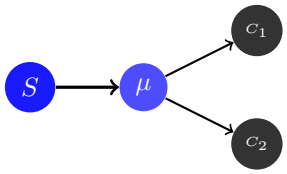

Fig. 1. The basic memory-assisted network flow compression scenario.

Next, we characterize the benefits of network memory in the context of a universal source coding problem for the class of smooth parametric sources [7], [10]. Let $l(C_n, x^n) = l_n(x^n)$ denote the length function that describes the codeword associated with the sequence x^n. The expected codeword length $\mathbf{E}l_n(X^n)$ quantifies the compression performance of UcompNmem, where $\mathbf{E}[\cdot]$ denotes the expectation operator. For an asymptotically optimal code in the sense that it achieves the entropy rate, we have $\frac{1}{n}(\mathbf{E}l_n(X^n) - H_n(X^n)) \to 0$ as $n \to \infty$, where $H_n(X^n)$ denotes the entropy of a sequence of length n. In the case that the router is capable of memorization (UcompWmem), assume that y^m is another sequence of length m from the same source that generates x^n. By context memorization, we mean that both the source S and the memory unit μ have already visited the sequence y^m. Let $l_{n|m}$ be a regular length function where S and μ have access to a context memory of length m from the previous communications. Then, the expected codeword length with memory $\mathbf{E}l_{n|m}(X^n)$ characterizes the compression performance of UcompWmem for x^n in the link from S to μ, in Fig 1.

Let $Q(l_n, l_{n|m})$ be defined as the ratio of the expected codeword length of UcompNmem to that of UcompWmem as

$$Q(l_n, l_{n|m}) \triangleq \frac{\mathbf{E}l_n(X^n)}{\mathbf{E}l_{n|m}(X^n)}. \tag{1}$$

Further, let ϵ be a real number such that $0 < \epsilon < 1$. We denote $g(n, m, \epsilon)$ as the fundamental gain of the context memorization on the family of parametric sources on a sequence of length n using a context sequence of length m for a fraction $1-\epsilon$ of the sources, which is defined as follows:

$$g(n, m, \epsilon) = \sup_{z \in \mathbb{R}} \left\{ z : \ \mathbf{P}\left[Q(l_n, l_{n|m}) \geq z \right] \geq 1 - \epsilon \right\}. \tag{2}$$

In other words, the fundamental gain of memorization is at least $g(n, m, \epsilon)$ for a fraction $1-\epsilon$ of the sources in the family.

In [10], Beirami and Fekri studied the fundamental limits of compression without memory. In [7], they extended their study to the behavior of the memorization gain $g(n, m, \epsilon)$ with respect to the sequence length and the memory size for different source models. In the rest of this paper, we investigate the effect of the context memorization gain on a network given that the memory-assisted compression gain $g(n, m, \epsilon)$ (hereafter, referred as g) is known.

III. Memory Deployment Problem in Networks

In this section, we will investigate the memorization gain on a network. A network is represented by an undirected graph $G(V, E)$ where V is the set of N nodes (vertices) and $E = \{uv : u, v \in V\}$ is the set of edges connecting nodes u and v. We consider a single source S which is the content server, and a set of memories $\boldsymbol{\mu} = \{\mu_i\}_{i=1}^{M}$ chosen out of N nodes. The content server is assumed to be a parametric information source [7]. We assume that each client requests a small to moderate length size sequence from the content server. As discussed in Sec. II, there is a fundamental limit beyond the entropy on the universal compression of a small to moderate length sequence. Therefore, UcompNmem will only be able to compress the content to a value which may be significantly larger than the entropy of the sequence. On the other hand, a memory nodes μ_i is capable of memorizing the communication between the server and some client node C_i.

To investigate the gain of memorization in the compression of the network flow, we must consider two phases. The first is the memorization phase in which we may assume all memory units have visited some sufficiently long sequence (or equivalently, a sufficient collection of small to moderate length sequences) from the source. This phase is realized in actual communication networks by observing the fact that a sufficient number of clients may have previously retrieved small to moderate length sequences from the server such that, via their routing, each of the memory units has been able to memorize the source. In the second phase, which is the subject of this section, we assume each node in the entire network may request (a small to moderate length) content from the server, uniformly. Our goal is to characterize the memorization gain in the compression of the sequences that are retrieved in the second phase. The above view simplifies our study as we are not concerned with the transition phase during which the memorization is taking place in the memory units. Hence, we can assume each memory unit will provide the same memory-assisted compression gain of g of the link from source to itself that depends on the length of the sequence that is being transmitted as well as the length of the sequence that is memorized in the memory unit, as described in Sec. II.

The goal of the memory deployment is to minimize the total cost of communication between the source and destinations in the network, measured by bit×hop, by deploying a set of memories $\boldsymbol{\mu}$. We wish to study the behavior of the total savings in terms of bit×hop, as a function of the number of the memories, place of memories, size of memories, length of the sequences, and the information source model.

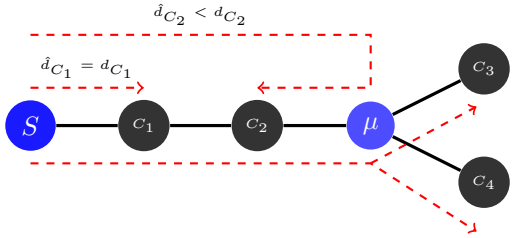

Fig. 2. The shortest walk between the source and destination is not necessarily a path when we have memory in network.

In a network with source S and a set of destinations $\mathbf{D} = \{D_i\}_{i=1}^{N}$, let f_D be the flow destined to $D \in \mathbf{D}$. The distance between any two nodes u and v is shown by $d(u, v)$. The distance is measured as the number of hops in the shortest path between two nodes. As we will see later, introducing memories to the network will change the lowest cost paths from the source to destinations, as there is a gain associated with the $S-\mu$ portion of the path. Therefore, we have to modify paths accounting for the gain of memories. Accordingly, for each destination D, we define *effective walk*, denoted by $W_D = \{S, u_1, \ldots, D\}$, which is the ordered set of nodes in the modified (lowest cost) walk between the source and D. Then, we partition the set of destinations as $\mathbf{D} = \mathbf{D}_1 \cup \mathbf{D}_2$, where $\mathbf{D}_1 = \{D_i : \exists \mu_{D_i} \in W_{D_i}\}$ is the set of destinations observing a memory in their effective walk, and $\mu_{D_i} = \arg\min_{\mu \in \boldsymbol{\mu}} \{\frac{d(S,\mu)}{g} + d(\mu, D_i)\}$. The total flow $\mathcal{F}$ is then defined as

$$\mathcal{F} = \begin{array}{l} \sum_{D_i \in \mathbf{D}_1} \left(\frac{f_{D_i}}{g} d(S, \mu_{D_i}) + f_{D_i} d(\mu_{D_i}, D_i) \right) + \\ \sum_{D_j \in \mathbf{D}_2} f_{D_j} d(S, D_j) \end{array}. \tag{3}$$

Using (3), we define $\hat{d}_D$, called the *effective distance* from source to D, as

$$\hat{d}_D = \begin{cases} \frac{d(S,\mu_D)}{g} + d(\mu_D, D) & D \in \mathbf{D}_1 \\ d(S, D) & D \in \mathbf{D}_2 \end{cases}. \tag{4}$$

In short, the effective distance is the distance when memory-assisted source compression is performed and hence the gain g applies. By definition, $\hat{d}_D \leq d(S, D)\ \forall D$. For simplicity, we assume $f_D = 1$ for all destinations. Hence, $\mathcal{F} = \sum_{D \in \mathbf{D}} \hat{d}_D$.

In a general network where every node can be a client, we define a generalized network-level gain of memory deployment as a function of memorization gain g, as follows:

$$\mathcal{G}(g) = \frac{\mathcal{F}_0}{\mathcal{F}}, \tag{5}$$

where $\mathcal{F}_0$ is the total flow in the network under UcompNmem, i.e., $\mathcal{F}_0 = \sum_{D \in \mathbf{D}} d(S, D)$.

In order to show the challenges of the memory deployment problem, we show as to how a single memory changes the effective paths in a network with a single source. Consider the network with the source node S placed as shown in Fig. 2.

The destinations are nodes $C_1, \ldots, C_4$, and $g = 4$. The effective walks from the source to destinations are obviously the shortest paths when there is no memorization (UcompNmem). As shown in the figure, the placement of memory changes the effective path to C_2 while the shortest paths from the source to C_1, C_3, and C_4 are the same as the effective paths. Without memory, the shortest path to C_2 is two hops long ($S \rightarrow C_1 \rightarrow C_2$), while the memory totally changes the effective walk distance to C_2 to $\hat{d}_{C_2} = \frac{3}{4} + 1$ as depicted in the figure.

In order to extend our study to analyze the achievable gains in general networks and study the behavior of $\mathcal{G}(g)$, in the next section, we consider the memory deployment gain in the network graphs that resemble the ER random graph [11]. The ER random graph is the building block of the recent models for complex graphs and hence the results would be useful in much broader contexts. We specifically direct our attention to connected random graphs since they better describe real networks.

IV. Gain of Memory Deployment on the Network Flow Compression

Definition 1: An ER random graph $G(N, p)$ is an undirected, unweighted graph on N vertices where every two vertices are connected with probability p.

Definition 2: Let $u, v \in G$ be any two vertices. The diameter of a connected graph is defined as $\max_{u,v} d(u, v)$. Similarly, the average distance of a connected graph is defined as $\mathbf{E}[d(u, v)]$.

The following properties hold for ER random graphs:

1) If $p < \frac{(1-\epsilon)\log N}{N}$, then $G(N, p)$ almost surely (a.s.) has isolated vertices and thus disconnected.
2) If $p = \frac{c \log N}{N}$ for some constant $c > 1$, then $G(N, p)$ is a.s. connected and every vertex asymptotically has degree $c \log N$ [12].
3) The diameter of $G(N, p)$ is almost surely $\frac{\log N}{\log Np}$.
4) The average distance in $G(N, p)$, denoted by $\bar{d}$, is

$$\bar{d} = (1 + o(1))\frac{\log N}{\log Np}, \tag{6}$$

provided that $\frac{\log N}{\log Np}$ goes to infinity as $N \rightarrow \infty$ (this condition is satisfied in the connected regime) [13].

Again, the main question is how $\mathcal{G}(g)$ scales with M. In order to characterize the gain of memory placement, we consider connected $G(N, p), p = \frac{c \log N}{N}$, with a single source node S and all other nodes as destinations. Since the expected degree of all nodes in ER graph is the same and every vertex is a destination with equal probability, we select memories $\{\mu_i\}_{i=1}^{M}$ uniformly and random. Theorem 1, below, provides the scaling of $\mathcal{G}(g)$ with respect to M:

Theorem 1: Suppose M is the number of deployed memories in an ER random graph. Let ϵ be a positive real number.

(a) If $M = O\left(N^{\frac{1}{g}-\epsilon}\right)$, then $\mathcal{G}(g) \sim 1$. [1]

(b) If $M = \Omega\left(N^{\frac{1}{g}+\epsilon}\right)$, then all the destinations benefit from memory and $\mathcal{G}(g) \overset{a.s.}{=} \frac{g}{1 - g\log_N(\frac{M}{N})}$.

Sketch of the proof: We first find an upper bound on the number of destinations benefit form each memory. This upper bound is sufficient to derive part (a) of the theorem. For the second part, we find a lower bound on the number of benefiting destinations. ∎

To characterize $\mathcal{G}(g)$, we first need to find $\mathcal{F}_0$. The average distance from the source to a node is $\bar{d}$. Thus, $\mathcal{F}_0 = N\bar{d}$. For large N, (6) results in $\mathcal{F}_0 \sim \frac{N \log N}{\log\log N}$.

Next, we need to find $\mathcal{F}$. For every memory μ we consider a neighbourhood $\mathbf{N}_r(\mu)$ as shown in Fig. 3. This neighborhood consist of all vertices v within distance r from μ. We choose r such that, almost surely, all nodes in $\mathbf{N}_r(\mu)$ would benefit from the memory μ. Clearly, if $\frac{d(S,\mu)}{g} + r = d(S, v)$, the benefit of the memory for node v vanishes and only nodes at distances less than r benefit from the memory μ. Given g, we denote this set of nodes benefiting from μ by $\mathbf{N}_r(\mu, g)$.

$$\mathbf{N}_r(\mu, g) = \left\{ v : \frac{d(S,\mu)}{g} + d(\mu, v) \leq d(S, v) \right\}. \tag{7}$$

Since memories are uniformly placed, the average value of $d(S, \mu)$ in $\hat{d}_v$ is equal to $\bar{d}$. Similarly, the average of $d(S, v)$ is also $\bar{d}$. Hence, solving for r in (7) and then using the result on the average distance in (6), we conclude

$$r \overset{a.s.}{=} (1 - 1/g)\left(\frac{\log N}{\log\log N}\right). \tag{8}$$

The following lemma, by Chung and Lu [13], gives an upper bound on the total number of vertices in the neighborhood $|\mathbf{N}_r(\mu_i, g)|$, where $|\cdot|$ is the set size operator.

Lemma 2 ([13]): Assume a connected random graph. Then, for any $\epsilon > 0$, with probability at least $1 - \frac{1}{(\log N)^2}$, we have $|\mathbf{N}_r(\mu_i, g)| \leq (1 + 2\epsilon)(Np)^r$, for $1 \leq r \leq \log N$.

Using Lemma 2 and (8), we deduce that

$$\begin{aligned} |\mathbf{N}_r(\mu_i, g)| &\overset{a.s.}{\leq} (1 + 2\epsilon)(\log N)^{(1-\frac{1}{g})\left(\frac{\log N}{\log\log N}\right)} \\ &= (1 + 2\epsilon)N^{1-1/g}. \end{aligned} \tag{9}$$

Therefore, the total number of nodes gaining from the memories is upper-bounded by $\sum_{i=1}^{M} |\mathbf{N}_r(\mu_i, g)| \leq M(1 + 2\epsilon)N^{1-1/g}$. As we will see, from (9) it is clear that the gain of memory vanishes if M is chosen small. The value $N^{1/g}$ is the threshold value for the network-wide gain. More accurately, if $M = O\left(N^{\frac{1}{g}-\epsilon}\right)$, there is no gain from memories.

Proof of Theorem 1(a): For all the nodes in $\mathbf{N}_r(\mu_i, g)$, we have a flow gain of g. Let $M = N^{\frac{1}{g}-\epsilon}$, then we have

$$\mathcal{G}(g) \leq \frac{N\bar{d}}{\frac{\bar{d}}{g}M|\mathbf{N}_r(\mu, g)| + \bar{d}(N - M|\mathbf{N}_r(\mu, g)|)} \tag{10}$$

$$\overset{a.s.}{\leq} \frac{N}{N - (1 - 1/g)MN^{(1-\frac{1}{g})}} \tag{11}$$

$$= \frac{N}{N - (1 - 1/g)N^{1-\epsilon}} \sim 1,$$

[1] In this paper, we have used the following asymptotic notation: $f(x) \sim g(x)$ iff $f(x)/g(x) \rightarrow 1$.

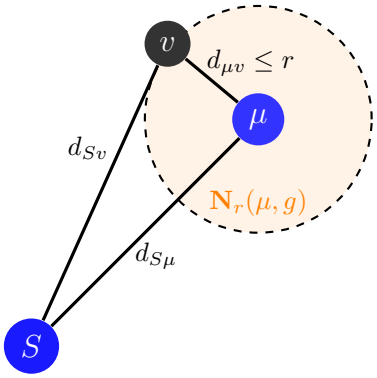

Fig. 3. Memory Neighborhood

where inequality in (10) follows from the double counting of the destination nodes that may reside in more than one neighborhood. Also, (11) follows from replacing (9) in (10). ■

Since we need more than $n^{\frac{1}{g}}$ memory units to have a network-wide gain, the next question is as to how $\mathcal{G}(g)$ scales when the number of memory units exceeds $n^{\frac{1}{g}}$. To answer this question, we need to establish a lower-bound on the neighborhood size and the number of nodes benefiting from memory. Further, we have to account for the possible double counting of the intersection between the memory neighborhoods. We use the following concentration inequality from [13] to establish the desired bound.

Proposition 3 ([13]): If $X_1, X_2, \ldots, X_n$ are non-negative independent random variables, then the sum $X = \sum_{i=1}^{n} X_i$ holds the bound

$$\mathbf{P}[X \leq \mathbf{E}[X] - \lambda] \leq \exp\left(-\frac{\lambda^2}{2\sum \mathbf{E}[X_i^2]}\right).$$

This inequality will be helpful to show that the quantities of interest concentrate around their expected values.

The following lemma provides a lower-bound on the neighborhood size $|\mathbf{N}_r(\mu, g)|$ and the lower-bound on $\mathcal{G}(g)$, as we show, is immediate.

Lemma 4: Consider a set of vertices V of $G(N, p)$ such that $\frac{|V|}{N} = o(1)$. For $0 < \epsilon < 1$, with probability at least $1 - e^{-Np|V|\epsilon^2/2}$, we have

$$|\mathbf{N}_r(\mu, g)| \geq (1 - \epsilon)(Np)^r. \tag{12}$$

Proof: The vertex boundary of V, denoted by $\Gamma(V)$, consists of all vertices in G adjacent to some vertex in V.

$$\Gamma(V) = \{u : u \notin V, \text{ and } u \text{ is adjacent to } v \in V\}.$$

Let X_u be the indicator random variable that a vertex u is in $\Gamma(V)$, i.e., $\mathbf{P}[X_u = 1] = \mathbf{P}[u \in \Gamma(V)]$. Then,

$$\begin{aligned}
\mathbf{E}\left[|\Gamma(V)|\right] &= \sum_{u \notin V} \mathbf{E}[X_u] = \sum_{u \notin V} \mathbf{P}[u \in \Gamma(V)] \\
&= \sum_{u \notin V} \left(1 - (1-p)^{|V|}\right) \\
&\geq p|V|(N - |V|) = (1 - o(1))Np|V| \quad (13)
\end{aligned}$$

where the inequality in (13) follows from

$$\mathbf{P}[u \in \Gamma(V)] = 1 - (1-p)^{|V|} \geq 1 - e^{-p|V|} \approx p|V|,$$

and the second part holds because $\frac{|V|}{N} = o(1)$. Since, X_u's are non-negative independent random variables, by applying Proposition 3 with $\lambda = \sqrt{\alpha \mathbf{E}[|\Gamma(V)|]}$, with probability at least $1 - e^{-\alpha/2}$ we have

$$\begin{aligned}
|\Gamma(V)| &\geq \mathbf{E}\left[|\Gamma(V)|\right] - \sqrt{\alpha \mathbf{E}[|\Gamma(V)|]} \\
&\geq (1 - \epsilon)Np|V|. \quad (14)
\end{aligned}$$

By picking a single vertex and applying (14) inductively r times, and then adding up the number of adjacent nodes, we obtain (12). ■

Now that we have a lower-bound on the number of nodes benefiting from each memory, we show that by increasing the number of memories beyond $M = N^{\frac{1}{g}}$, memories cover all the nodes in the graph effectively and hence all the nodes would gain from the memory placement.

In order to limit the intersection between the neighborhoods, we reduce r to r_δ as below:

$$r_\delta = (1 - 1/g - \delta)\left(\frac{\log N}{\log\log N}\right). \tag{15}$$

With this choice of r_δ, by lemmas 2 and 4, we deduce that the probability that a random node $u \in G$ belongs to the neighborhood $\mathbf{N}_{r_\delta}(\mu_i, g)$ of the memory μ_i is $N^{-1/g-\delta}$. Hence, the expected number of the covered nodes is

$$\begin{aligned}
\mathbf{E}\left[\left|\bigcup_{i=1}^{M} \mathbf{N}_{r_\delta}(\mu_i, g)\right|\right] &= \sum_{u \in G} \mathbf{P}\left[u \in \cup_{i=1}^{M} \mathbf{N}_{r_\delta}(\mu_i, g)\right] \\
&= \sum_{u \in G} \left(1 - (1 - N^{-1/g-\delta})^M\right) \\
&\approx N\left(MN^{-1/g-\delta}\right) = N, \quad (16)
\end{aligned}$$

where (16) holds by choosing $M = N^{1/g+\delta}$.

To show that the number of covered nodes is concentrated around its mean, we use Prop. 3 again with $\lambda = \sqrt{\alpha \mathbf{E}\left[|\cup \mathbf{N}_{r_\delta}(\mu_i, g)|\right]}$. Then, with probability at least $1 - e^{-\alpha/2}$ we have

$$\begin{aligned}
\left|\bigcup_{i=1}^{M} \mathbf{N}_{r_\delta}(\mu_i, g)\right| &\geq \mathbf{E}\left[|\cup \mathbf{N}_{r_\delta}(\mu_i, g)|\right] - \lambda \\
&\geq (1 - o(1))N.
\end{aligned}$$

Hence, the memories cover, almost surely, all of the nodes.

Since all nodes are covered with high probability, we can associate each node with a neighborhood $|\mathbf{N}_{r_\delta}(\mu_i, g)|$, for which nodes' distances in the neighborhood from memory are $(1 - o(1))r_\delta$.

Proof of Theorem 1(b): By (16), we can bound the network-wide gain of the memory from below. We have

$$\begin{aligned}
\mathcal{G}(g) &\overset{a.s.}{=} \frac{N\bar{d}}{(\bar{d}/g + r_\delta)N} \quad (17) \\
&= \frac{1}{1/g + (1 - 1/g - \delta)} = \frac{1}{1 - \delta}, \quad (18)
\end{aligned}$$

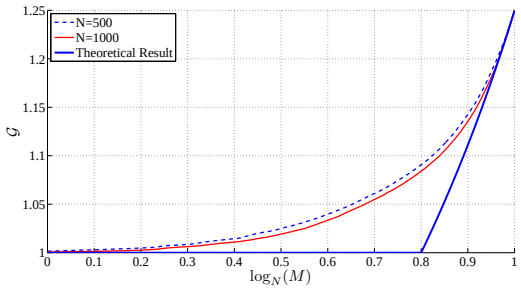

Fig. 4. Network-level gain $\mathcal{G}$ versus $\log_N(M)$ for different network sizes N and $g = 1.25$.

where (17) holds because the distance of the nodes from memory is r_δ, asymptotically almost surely. ■

As the number of memories becomes close to N, i.e., $\delta \to (1 - \frac{1}{g})$, the gain $\mathcal{G} \to g$, as expected. In the next section, we verify our result in memory-assisted source coding and the network-wide gain of memory via numerical simulations.

V. Simulation Results

In this section, we first demonstrate our theoretical results through an example. We consider the source to be a first-order Markov source with alphabet size equal to 256. In [7], Beirami and Fekri derived a lower bound on the memorization gain as a function of the sequence length and the memory size. For example, they showed that $g(512\text{kB}, 8\text{MB}, 0.05)$ is about 1.25, i.e., with a memory of 8MB, a gain of 1.25 is obtained on the memory-assisted compression of 512kB long sequences [7]. Fig. 4 presents the simulation results for network-wide gain for different network sizes versus $\log_N(M)$ when $g = 1.25$. The rightmost solid curve is our theoretical result in (17). For small values of M, the network-wide gain would be 1 for $N \to \infty$, while for large M, $\mathcal{G}$ tends to g. Also, as N increases, simulation results approach the theoretical limit for both small and large values of M.

References

[1] Z. Zhuang, C.-L. Tsao, and R. Sivakumar, "Curing the amnesia: Network memory for the internet," Tech Report, 2009. [Online]. Available: http://www.ece.gatech.edu/research/GNAN/archive/tr-nm.pdf

[2] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren, "Cloud control with distributed rate limiting," in *SIGCOMM*. ACM, 2007, pp. 337–348.

[3] A. Anand, V. Sekar, and A. Akella, "Smartre: an architecture for coordinated network-wide redundancy elimination," *SIGCOMM*, vol. 39, no. 4, pp. 87–98, 2009.

[4] Z. Zhuang, T.-Y. Chang, R. Sivakumar, and A. Velayutham, "Application-aware acceleration for wireless data networks: Design elements and prototype implementation," *IEEE Trans. Mobile Computing*, vol. 8, no. 9, pp. 1280 –1295, sep. 2009.

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

[6] F. Willems, Y. Shtarkov, and T. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Trans. Info. Theory*, vol. 41, no. 3, pp. 653–664, May 1995.

[7] A. Beirami and F. Fekri, "On the performance of universal source coding for finite-length sequences," *submitted to IEEE Trans. Info. Theory*.

[8] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Info. Theory*, vol. 19, pp. 471–480, 1973.

[9] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," *Proc. IEEE Data Compression Conference*, pp. 158–167, March 1999.

[10] A. Beirami and F. Fekri, "Results on the redundancy of universal compression for finite-length sequences," in *IEEE Intl. Symp. Info. Theory (ISIT)*, 2011, pp. 1604–1608.

[11] P. Erdős and A. Rényi, "On random graphs. I." *Publicationes Mathematicae*, pp. 290–297, 1959.

[12] N. Alon and J. Spencer, *The Probabilistic Method–3rd edition*. John Wiley & Sons, USA, 2008.

[13] F. Chung and L. Lu, *Complex Graphs and Networks*. American Mathematical Society, 2006.