Packet-Level Clustering for Memory-Assisted Compression of Network Packets

Liling Huang,[1] Ahmad Beirami,[2] Mohsen Sardari,[3] Faramarz Fekri,[3] Bo Liu,[1] Lin Gui[1]

[1]Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China

[2]Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA

[3]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

Email: [1]{sunny_hll,liubo_lb,guilin}@sjtu.edu.cn, [2]a.beirami@duke.edu, [3]{mohsen.sardari, fekri}@ece.gatech.edu

***Abstract*—With the explosive growth of the Internet traffic, data compression can be a powerful technique to improve the efficiency of data transfer in networks and consequently reduce the cost associated with the transmission of such data. Recently, we proposed a memory-assisted compression framework that utilizes the packet-level memorized context to reduce the inevitable redundancy in the universal compression of the payloads of the short-length network packets. In this paper, we investigate the practical aspects of implementing cluster-based memory-assisted compression and proposed a non-parametric clustering algorithm for training packet selection. We demonstrate that, when compression speed is not an issue, our proposed non-parametric clustering algorithm with Lite PAQ compression algorithm can achieve nearly 70% traffic reduction on real data gathered from Internet traffic. We also explore the trade-off between the memory-assisted compression speed and performance using different clustering algorithms and compression methods.**

***Index Terms*—Memory-Assisted Compression; Networking; Redundancy Elimination; Non-parametric Clustering.**

I. Introduction

The very high amount of data traffic produced and transmitted daily around the world call for new techniques to reduce the considerable amount of redundancy in the traffic. Recent studies confirm that most of this redundancy is present at the packet level [1], [2]. In other words, packets generated by the same or different sources and destined to the same or different clients contain significant cross-packet correlation. Universal compression has undergone a lot of developments in the past few decades [3]–[10]. However, for an IP packet with a length only approximately 1500 bytes, the state-of-the-art compression techniques, e.g., context tree weighting (CTW) [11], [12], are inefficient in capturing the redundancy in data as compression performance primarily depends on the sequence length (cf. [7], [13] and the references therein). In other words, there is a significant penalty with respect to what is fundamentally achievable when we attempt to universally compress a finite-length packet [13]. On the other hand, many of the packets share common context that can be exploited for better compression. It is very desirable to encode and compress each individual delivering packet more efficiently by utilizing the dependency across multiple packets and side-information provided from the memory. Therefore, a protocol-independent and content-aware network packet compression scheme is required for removing the data redundancy.

This work was carried out when L. Huang and A. Beirami were affiliated with Georgia Institute of Technology. The work of A. Beirami, M. Sardari, and F. Fekri was supported by the National Science Foundation under Grant No. CNS-1017234.

Thus far, many researchers have investigated the opportunities for eliminating the redundancy in the network packets by equipping some nodes in the network with memorization capability in order to perform better packet-level redundancy elimination via deduplication (cf. [14]–[18]). However, there is even more space to improve beyond the simple deduplication if we look into the statistical redundancies within a packet as well as significant dependencies that exist across packets. These statistical redundancies can potentially be suppressed using statistical compression techniques.

In [1], [19], we developed a framework for compression of small sequences, called *memory-assisted compression* and introduced its potential application in network compression. In memory-assisted framework, compression of a sequence is performed using a memory of the previously seen sequences. Consequently, every sequence can be compressed far better compared to the case that the sequence is compressed on its own without considering the memory. We showed that memory-assisted compression has significant potential for removal of the redundancy from network packets. To realize memory-assisted compression in the network, in [2], we investigated clustering for data compression and proposed a clustering algorithm based on Hellinger distance metric in the context of memory-assisted compression. In [20], we further provided theoretical justification as to why clustering is optimal in the context of memory-assisted compression. However, it remained an open problem as to provide the optimal combination of clustering algorithm and compression method for memory-assisted compression that works efficiently in practice. In this paper, we explore the combination of different compression methods as well as clustering algorithms and describe the benefits of each for a given application. In particular, we present an effective non-parametric clustering algorithms for memory-assisted compression which eliminates the need for knowing the number of clusters in advance.

The rest of this paper is organized as follows. In Section II, we present the necessary background on universal compression. In Section III, we provide an applicable module framework and key design elements for memory-assisted data compression with clustering. In Section IV, we discuss the feature extraction and distance metric for network packets. In Section IV, we propose a non-parametric clustering algorithm for training packets selection from common memory. In Section V, we provide simulation with real network traces with different schemes demonstrating the effectiveness of the proposed non-parametric clustering. Finally, Section VI concludes this paper.

II. Background Review and Related Work

In this section, we first briefly review the necessary background and previous work on the memory-assisted compression. Let a parametric source be defined using a d-dimensional parameter vector $\mu_\theta = (\mu_1, ..., \mu_d) \in \Delta$ that is a priori unknown, where d denotes the number of the source parameters and $\theta = (\theta^{(1)}, ..., \theta^{(K)})$ is the K-source mixture randomly generated from the source parameter vector space Δ.

Let $X^n = \{x_1, x_2, \ldots, x_n\}$ denote sample packet with length n from the mixture source model θ. Denote $\mathbf{Y}^{n,T} = \{y^n(t)\}_{t=1}^{T}$ as the set of the previous T sequences shared between compressor and decompressor, where $y^n(t)$ is a sequence of length n generated from the source $\theta^{P(t)}$. In other words, $y^n(t) \sim \theta^{(P(t))}$. Further, denote $\mathbf{P}$ as the vector $\mathbf{P} = (P(1), ..., P(T))$, which contains the indices of the sources that generated the T previous side information sequences.

Most of the universal compression schemes are strictly lossless codes, namely, Lempel-Ziv [21], [22], CTW algorithm [11] and Lite PAQ algorithm [23]. We assume that, in Fig. 1, both the compressor and the decompressor have access to a common side information of previous packet sequences $\mathbf{y}^{n,T}$ from the mixture of K parametric sources, where each packet is independently generated according to the above procedure. We discuss the following four data preprocessing schemes of compression performance of sample packet X^n in the following.

- Ucomp: Universal compression, which is the conventional compression based solution without utilizing memory.
- UcompSM: Universal compression with simple memory (common memory between the compressor and the decompressor), which treats the side information as if it were generated from a single parametric source.
- UcompPCM: Universal compression with perfectly clustered memory (based on the source indices), which assumes that the source indices of the side information sequences are labeled, and hence, only the relevant side information is used toward the compression of a new packet.[1]
- UcompCM: Universal compression with clustered memory. This utilizes the non-parametric clustering-based scheme to select useful memory, which will be described in Section IV.

[1]Please note that UcompPCM scheme is not practically interesting as the source indices $\mathbf{P}$ of the sequence is usually not available.

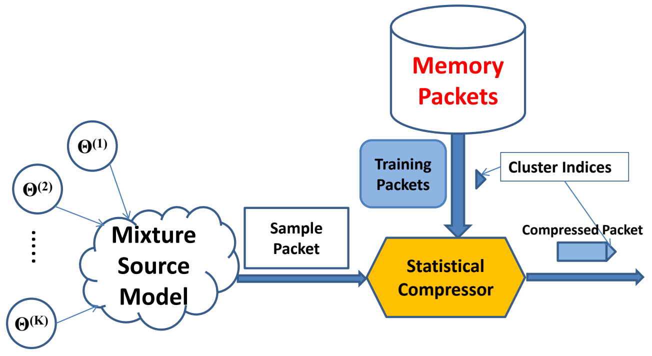

Fig. 1. Architecture of the memory-assisted compression module with clustering at data source.

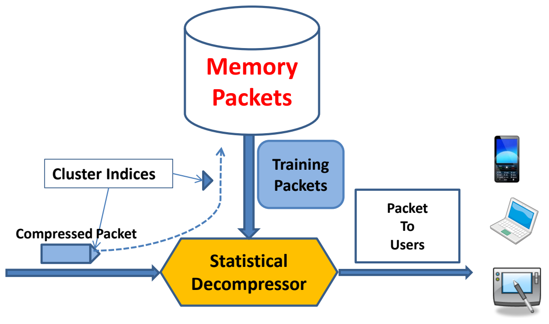

Fig. 2. Data decompression module architecture at the last hop node.

In the case of universal compression, the single packet alone without knowing any memory. The expected code lengths is given in [20] by

$$H(X^n) = H_n(\theta) + \frac{d}{2}\log n + O(1). \quad (1)$$

In case of universal compression with clustered side information, the compression is trained with packets from the same source. In [20], the conditional entropy of X^n is given by

$$H(X^n|\mathbf{Y}^{n,T}) = H_n(\theta) + \frac{d}{2}\log\left(1+\frac{nK}{m}\right) + O\left(\frac{1}{n}+\frac{1}{\sqrt{T}}\right), \quad (2)$$

where $H_n(\theta)$ is the entropy of source model θ. The compressor uses a memory of size $m = nT$ (T packets of size n) to compress a new sequence of length n from X. This indicates the optimal compression rate for compression of packet with T/K training packets from the same one source model.

Hence, when T is sufficiently large, we expect that UcompPCM has the same performance as (2) indicates. This indeed demonstrates that *clustering* is optimal for the universal compression with side information [20]. As such, we pursue the clustering of the side information (i.e., memory) in Section IV.

III. Memory-assisted Data Compression Module with Clustering

This section elaborates the detailed design of 3.5 layer data compression system, which implements the memory assisted compression of real data transmission between data source and data destination. We first describe system deployment and motivational scenarios, and then discuss the details of the system.

A. Module Deployment

As we know for traditional five-layer architecture of cellular core network, our memory-assisted data compression system with clustering is a 3.5 layer module deployed between the network layer and transport layer. At network layer, the compressor at the content generator server extracts IP payload as input, and encapsule compressed content to IP packet again before sending to next hop. This compression system mainly compresses the IP packets with lossless coding algorithm and decompresses the packets at the last-hop switches or routers prior to the final destination node, improving transmission efficiency of down-link data. By saving volume of packets to be sent, large redundancy in networks can be reduced. Taking General Packet Radio Service (GPRS) cellular network as an example, the Serving GPRS Support Node (SGSN) pool works as data source, while the Base Station is taken as data destination. Another example is the backbone network of Google's network infrastructure, compression can also reduce redundancy in traffic from data center to Edge Point of Presence. Please note that data transmission in the backhaul comprises a large chunk of Google's overall costs.

B. Module Architecture and Operations

Fig. 1 shows the data source architecture. The compression at data source is divided into two stages, which are named offline clustering stage and online training selection stage. First, common memory packets are partitioned into different clusters for maximum compression performance. Second, for every new sample packet, memory packets sharing similar statistical properties are clustered together as a side information for compression of sample packet (referred as training selection). After compression with proper information, the compressor returns both the indices of clusters and the codeword for the compressed sequence as the new payload of IP packets.

Fig. 2 illustrates the data decompression operations. The data destination accomplishes the procedures of compression in reverse order. With access to the common memory and the indices of associated clusters given by received compressed packet, the decompressor recovers the original content by indexing training clusters and distributes it to receivers according to IP header.

C. Key Concerns for Implementation

The proposed memory-assisted compression uses data clustering to accomplish selection of useful training data. Before delving into the details of the algorithms, we need to discuss the key problems to be solved to achieve the optimal compression performance in practice. These will be discussed in detail in the future sections of the paper.

1) Training Packet Selection Algorithm: Theoretically, the optimal training set for compressor is the combination of packets generated from the same source model [20]. In other words, we need to select the cluster that contains the packets from the same source model as the sample packet to be compressed. This requires dividing the data packets into groups (clusters) so as to be able to assign each new sample packet to the optimal cluster for efficient compression.

2) Selection of the Compression Method: Traditionally, dictionary-based compressors are applied to compress data in telecommunication systems for their high speed in online computation. Statistical compression has superior performance but has a slower speed. We comprehensively compare the performance of both compressors with different schemes in section V.

IV. Training Packet Selection Algorithm

In this section, we present two algorithms to accomplish packet-level clustering and selection of training packets to be fed to the compressor for memory-assisted compression. Before the introduction of algorithms, we firstly describe the infinite mixture source model and related aspects of training packet selection, including feature extraction and distance metric.

In real world networks, the number of sources in mixture model is not generally known. In this paper, we relax the assumption that we have the priori information about the clusters of the mixture source that was made in [2]. To address this problem, an infinite mixture model is used to represent the data sequences as

$$p\left(z|\Delta\right) = \lim_{m\to\infty} \sum_{i=1}^{m} w_i \delta\left(z-\theta^{(i)}\right), \tag{3}$$

where w_i is the probability that the packet is generated by source model $\theta^{(i)}$ in the mixture, and m denotes the number of all possible data models from source set Δ.

Feature extraction deals with generating simple descriptions for a large amount of data that can accurately describe characteristics of original data. In networks, characters are encoded via 8 bits or a multiple of 8 bits binary sequences (i.e., byte granularity). To deal with mixture source model, we choose a 256-symbol set as the minimum character unit for cluster analysis.

To measure the similarity between the packets, we select the Hellinger distance metric to quantify the similarity between two probability distributions. For any two discrete probability distributions P and Q, Hellinger distance is given by

$$H\left(P,Q\right) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{255}\left(\sqrt{P_i}-\sqrt{Q_i}\right)^2}, \tag{4}$$

Please note that Hellinger distance is a generalized form of the relative entropy (which is not a distance metric) and naturally coincides with the redundancy in the universal compression, which is our performance metric of interest.

A. k-Means Clustering Algorithm

Suppose the total number of clusters is given by K. Then, k-Means [24] clustering algorithm partitions the memory into K clusters by trying to minimize the total distance of the packets inside the clusters from their respective cluster centers. Then, the cluster with shortest distance from the sample packet X_n is assigned as the training for compression. Please refer to [2] for a detailed explanation of the clustering and classification using k-Means algorithm.

There are two major drawbacks in compression using k-Means clustering algorithm. First, the prior parameter of mixture source model needs to be known. Second, the performance and speed of the compressor both suffer from the large scale classified training set.

B. Non-Parametric Clustering Algorithm

K-means clustering algorithm can successfully cluster data packets in the ideal situation with static number of source model mixture. However, this algorithm can break down when the number of users are not able to predict the number of clusters correctly, especially for the infinite mixture source model in real world networks.

To cluster packets without knowing prior parameter of mixture source model , we propose a dynamic non-parametric clustering method. We partition memory into m small sub-clusters $S = \{s_1, \ldots, s_m\}$. Each sub-cluster consists of about T/m neighboring packets with the minimum variance.

Algorithm 1 Non-Parametric Clustering Algorithm

```
Compute empirical PDF vectors $\{d_i\}$
Compute sub-clusters $S = \{s_1, \ldots, s_m\}$
for Incoming packet $X^n$ do
    Compute distance $||X^n - s_i||$
    Current sub-cluster set $C = S$
    while $training\_pkt\_num < min\_training\_num$ do
        if $s_{closest_i} = min_{s_i \in C} ||X^n - s_i||$ then
            Training set $Q = Q \cup \{s_{closest_i}\}$
            Index set $T = T \cup \{closest_i\}$
            $training\_pkt\_num$ update
            Remove $s_{closest_i}$ from $C = \{s_1, \ldots, s_m\}$
        end if
    end while
    Return $Q$ and $T$
end for
```

As soon as the fine-grain sub-clusters are produced, the construction of the training packet set can be under process. Each sub-cluster is represented by the mean value of the included vectors, the similarity between sample packet X^n and sub-clusters is measured by the distance between X^n and s_i. After the initialization of the current sub-cluster set $C = S$, the sub-cluster from set C nearest to X^n is merged into the training set Q and is removed from C after merging. In other words, the new dynamic training set Q is updated by $P = \{s_{\pi_1}, \ldots, s_{\pi_K}\}$, where

$$\{\pi_1, \ldots, \pi_K\} = \arg \min_{\{\pi_1,\ldots,\pi_K\} \subset \{1,\ldots,T\}} \sum_{i=1}^{K} ||X^n - s_{\pi_i}|| \quad (5)$$

and $\pi_i \neq \pi_j$ if $i \neq j$. Also, $||X^n - s_{\pi_i}||$ represents the distance between X^n and s_{π_i}. The merging ends when the expected number of training packets is reached. The actual number of sub-clusters is fixed according to the minimum number of packets requirement of compressor. Algorithm 1 elaborates the procedures of the non-parametric clustering for selection of training packets.

In the real world case, when the feature space of data packets scattered in a high dimensional(255-dimension) space, the shapes of clusters are arbitrary, with varying densities. Especially, when the sample data packet is among none of the clusters, the performance of K-means clustering will be badly influenced. This non-parametric clustering algorithm take this characteristic of real world data as the basic assumption. By merging sub-clusters nearby, it can effectively collect most useful training data with appropriate consistency for sample packet compression. Besides, without giving the accurate number of clusters in advance, the non-parametric clustering algorithm achieves impressive improvement compared to K-means clustering. All the detailed simulation in next session will elaborate the performance of the proposed algorithm.

V. Simulation and Evaluation

In this section, we present simulation results to demonstrate the performance of the proposed memory-assisted compression system with non-parametric clustering and the overall improvement of memory-assisted compression over universal compression without memory. Furthermore, we discuss the trade-off between compression speed and performance.

A. Simulations on Real Network Traces

For a realistic evaluation, we perform simulation with data gathered from 20 different mobile users network traces in real world. The data set was gathered by S. Sanadhya *et al.* in [17]. First, we randomly generate packet sequences from the 27000-packet mixture of 15 users to construct the commonly accessible memory for clustering. Then, 10 sample packets from each of the 20 users (200 packets in total) are selected as test packets. Note the test packets are distinct from the packets used for training. Besides, there are 50 test packets that are generated from the 5 users which are not used for the generation of the training packets and hence do not have packets in the mixture memory. Average compression rate of each test packet is taken as the compression performance metric. Please note that each test packet is compressed separately. This is due to the packets flow in networks is a combination of packets from different sources and can not be simply compressed together. The whole simulation setup is summarized in table I.

B. Packet Selection Algorithm Performance Comparison

To compare the overall performance of memory-assisted compression with different clustering schemes, simulation results are summarized in Table II.

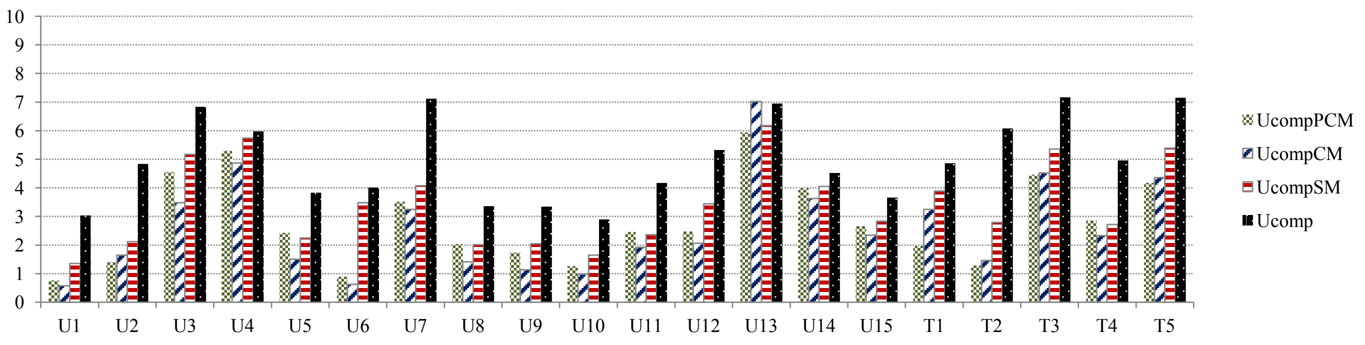

Fig. 3. Average compression-rate of Lite PAQ on real traffic data.

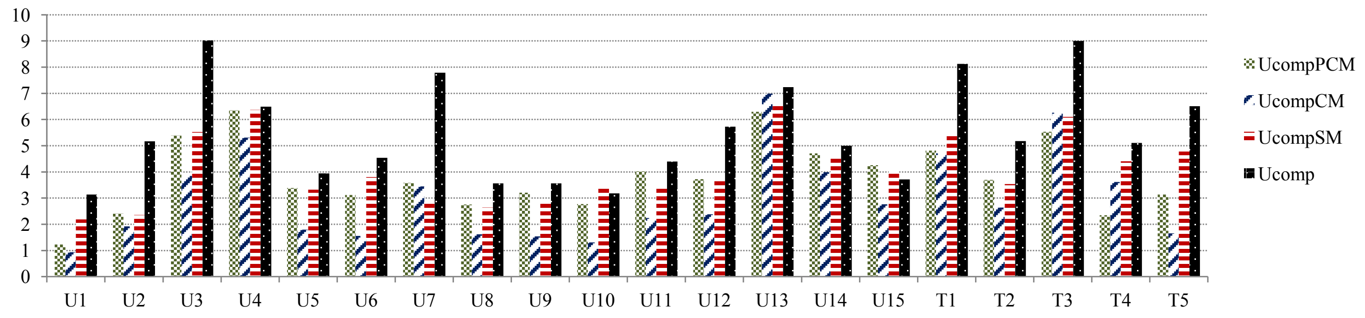

Fig. 4. Average compression-rate of CTW on real traffic data.

TABLE I
SIMULATION SETUP SUMMARY

Case	Value
Total Number of Users	20
Number of Users in Mixture Memory	15
Number Packets from Each User Data in Memory	1800
Total Number of Memory Data Packets	27000
Average Size of Each Data Packet	1KB
Approximate Size of Total Memory	25 MB
Number of Users for Performance Testing	20
Total Number of Test Packets	200
Distance Metric	Hellinger Distance
Clustering Algorithm	K-means, Non-parametric
Compression Algorithm	Lite PAQ, CTW, Gzip

We choose packet selection with two algorithms, namely, k-Means clustering algorithm and non-parametric clustering algorithm. According to Table II, non-parametric clustering achieve very similar performance, around 8% better than k-Means clustering. Besides, non-parametric clustering does not require to know the number of clusters in advance like k-Means clustering. By using ball tree data structure [25], the computational cost of nearest sub-clusters search is $O(N\log(N))$, where N is the number of sub-clusters. The average size of training packets selected by K-means clustering is around 1800 packets whereas around 200 packets by non-parametric clustering. With smaller-sized training packet selected by non-parametric clustering algorithm, the compression speed is 9 times quicker than that of K-means clustering. As the average size of clusters generated from K-means is 9 times larger thank the non-parametric counterpart.

TABLE II
AVERAGE COMPRESSION RATE WITH DIFFERENT CLUSTERING SCHEMES

Ratios (bits/byte)	k-Means	Non-parametric
Hellinger	2.63	2.42

Through comparison of performance of both clustering algorithm, proposed non-parametric clustering algorithm is proved to be more effective in network traffic redundancy reduction than referenced K-means clustering algorithm for real world data.

C. Memory-Assisted Compression Performance Evaluation

To demonstrate the impact of the side information on the compression performance, we analyze the average codeword length of the four important schemes (UcompPCM, Ucom-

TABLE III
THE AVERAGE TRAFFIC REDUCTION (%) OF DIFFERENT COMPRESSION SCHEMES ON THE REAL NETWORK TRAFFIC TRACES.

Avg. Traf. Red.	UcompPCM	UcompCM	UcompSM	Ucomp
Lite PAQ	65.47 %	69.62 %	59.37 %	41.77 %
CTW	52.36 %	65.32 %	51.83 %	36.29 %
Gzip	44.80 %	60.79 %	38.12 %	24.87 %

pCM, UcompSM, and Ucomp) discussed in Section II. Fig. 3 and Fig. 4 illustrate the average compression-rate on these data with Lite PAQ compressor and CTW compressor, respectively.

As can be seen, universal compression without help of any memory packets (Ucomp) results in longest average code lengths which verifies the penalty of finite-length compression described in [13]. UcompCM, which is the cluster-based memory-assisted compression, consistently outperforms all other schemes. It is worth noting that for the data from users which are not necessarily from mixture source model (user T1,..., T5), non-parametric clustering still achieves impressive improvement compared to simple memory assisted compression UcompSM. Compression with memory of user's previous packets UcompPCM sometimes performs well while it sometimes performs poorly due to the fact that the user data possibly comes from variant source models. In general, clustering algorithm is applicable to both Lite PAQ compression and CTW compression with impressive improvement.

Table III presents the average traffic reduction over all the fifteen users with different compression algorithms. Using the non-parametric clustering scheme, we compare the overall improvement of both dictionary-based compressor (Gzip) [26] and statistical compressor (Lite PAQ and CTW). As can be seen, Lite PAQ (which is close to the state-of-the art in compression) achieves nearly 70% traffic reduction and CTW achieves 65% reduction. With more than 65% traffic reduction, statistical compression outperforms dictionary-based compression, which offers 60% reduction. However, dictionary-based compression tends to have ten times higher compression speed. Wireless applications tolerate more latency compared to the wired networks. Hence, statistical compression is more suitable for wireless data compression while dictionary-based compression is likely to be employed in wired networks.

VI. CONCLUSION

In this paper, the clustering-based memory-assisted compression system is investigated. Via experimentation on real traces, we show that our proposed non-parametric clustering memory-assisted compression with Lite PAQ algorithm is indeed the most effective method among all clustering-based memory-assisted compression schemes, achieving nearly 70% traffic reduction in real network data. Finally from comparison with Gzip compression, we discuss the trade-off between compression speed and performance according to the characteristics of networks.

REFERENCES

[1] M. Sardari, A. Beirami, and F. Fekri, "Memory-assisted universal compression of network flows," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 91–99.

[2] M. Sardari, A. Beirami, J. Zou, and F. Fekri, "Content-aware network data compression using joint memorization and clustering," in *IEEE INFOCOM*, Turin, Italy, April 2013.

[3] L. Davisson, "Universal noiseless coding," *IEEE Trans. Info. Theory*, vol. 19, no. 6, pp. 783 – 795, nov. 1973.

[4] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. Info. Theory*, vol. 30, no. 4, pp. 629 – 636, jul. 1984.

[5] B. Clarke and A. Barron, "Information-theoretic asymptotics of bayes methods," *IEEE Trans. Info. Theory*, vol. 36, no. 3, pp. 453 –471, may. 1990.

[6] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Trans. Info. Theory*, vol. 44, no. 6, pp. 2743 –2760, oct. 1998.

[7] N. Merhav and M. Feder, "A strong version of the redundancy-capacity theorem of universal coding," *IEEE Trans. Info. Theory*, vol. 41, no. 3, pp. 714 –722, May 1995.

[8] D. Baron and Y. Bresler, "An O(N) semipredictive universal encoder via the BWT," *IEEE Trans. Info. Theory*, vol. 50, no. 5, pp. 928–937, May 2004.

[9] M. Effros, K. Visweswariah, S. Kulkarni, and S. Verdu, "Universal lossless source coding with the Burrows Wheeler transform ," *IEEE Trans. Info. Theory*, vol. 48, no. 5, pp. 1061–1081, May 2002.

[10] N. Krishnan, D. Baron, and M. K. Mihcak, "A parallel two-pass MDL context tree algorithm for universal source coding," in *2014 IEEE International Symposium on Information Theory Proceedings (ISIT '14)*, Jul. 2014.

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

[12] F. Willems, "The context-tree weighting method: extensions," *IEEE Trans. Info. Theory*, vol. 44, no. 2, pp. 792–798, Mar 1998.

[13] A. Beirami and F. Fekri, "Results on the redundancy of universal compression for finite-length sequences," in *IEEE Intl. Symp. Info. Theory (ISIT)*. IEEE, 2011, pp. 1504–1508.

[14] S. Hsiang-Shen, A. Gember, A. Anand, and A. Akella., "Refactoring content overhearing to improve wireless performance," in *MobiCom*, Las Vegas, NV, 2011.

[15] 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.

[16] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," *SIGCOMM*, vol. 38, pp. 219–230, 2008.

[17] S. Sanadhya, R. Sivakumar, K.-H. Kim, P. Congdon, S. Lakshmanan, and J. Singh, "Asymmetric caching: Improved deduplication for mobile devices," in *Proceedings of the ACM MOBICOM 2012 conference*. ACM, 2012.

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

[19] A. Beirami, M. Sardari, and F. Fekri, "Results on the fundamental gain of memory-assisted universal source coding," in *2012 IEEE International Symposium on Information Theory (ISIT '2012)*, July 2012, pp. 1092–1096.

[20] A. Beirami, L. Huang, M. Sardari, and F. Fekri, "On optimality of data clustering for packet-level memory-assisted compression of network traffic," in *15th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC 2014)*, Toronto, Candada, June 2014.

[21] 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.

[22] ——, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Info. Theory*, vol. 24, no. 5, pp. 530–536, Sep 1978.

[23] M. Mahoney, "Adaptive weighing of context models for lossless data compression," Florida Tech., Tech. Rep., 2005.

[24] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[25] S. M. Omohundro, *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[26] L. P. Deutsch, "Gzip file format specification version 4.3," 1996.