On Protocol-Independent Data Redundancy Elimination

On Protocol-Independent Data Redundancy Elimination

Yan Zhang, *Student Member, IEEE,* and Nirwan Ansari, *Fellow, IEEE*

***Abstract*—Data redundancy elimination (DRE), also known as data de-duplication, reduces the amount of data to be transferred or stored by identifying and eliminating both intra-object and inter-object duplicated data elements with a reference or pointer to the unique data copy. Large scale trace-driven studies have showed that packet-level DRE techniques can achieve 15-60% bandwidth savings when deployed at access links of the service providers, up to almost 50% bandwidth savings in Wi-Fi networks and as much as 60% mobile data volume reduction in cellular networks. In this paper, we survey the state-of-the-art protocol-independent redundancy elimination techniques. We overview the system architecture and main processing of protocol-independent DRE techniques, followed by discussion on major mechanisms activated in protocol-independent DRE, including the fingerprinting mechanism, cache management mechanism, chunk matching mechanism, and decoding error recovery mechanism. We also present several redundancy elimination systems deployed in wireline, wireless and cellular networks, respectively. Several other techniques to enhance the DRE performance are further discussed, such as DRE bypass techniques, non-uniform sampling, and chunk overlap.**

***Index Terms*—Data redundancy elimination (DRE), protocol-independent DRE, data de-duplication, content delivery acceleration, wide area network (WAN) optimization.**

I. Introduction

A significant amount of redundant traffic has long been observed over the communication networks [1–4]. The observed redundancy in Internet traffic is typically in the range of 15-60%. Internet traffic redundancy stems naturally from the distribution of highly-popular Internet contents and the large number of Internet users. Some of the contents on the Internet are highly popular objects which are requested and transferred repeatedly across the network for a large number of users. Moreover, this redundancy arises from common end-user activities, e.g., repeatedly accessing, retrieving, distributing the same or similar contents over the Internet several times a day. As a result, a large amount of the same or similar contents have been transferred repeatedly across the Internet in both client-server and peer-to-peer applications.

Redundant traffic wastes network resources, worsens the communication performance by saturating the network bandwidth, and increases the economic costs if usage-based charges are used. With the rapid growth of the Internet traffic [5], redundancy elimination has attracted much attention in recent years from the academia [1–4, 6–15] and industries, such as Cisco [16], Juniper [17], BlueCoat [18], and Riverbed [19]. A number of existing diverse systems and solutions have been explored to improve the network efficiency and communication performance by eliminating the redundant transfers in the network. It has been widely agreed that redundancy elimination offers significant benefits in practice. The overall benefit of data redundancy elimination is better network delivery performance in terms of higher network throughput, lower response time and higher effective network utilization.

A majority of traditional redundant elimination solutions operate at the application layer and object level, such as data compression [20] (e.g., GZip) which can remove the redundant content within one object efficiently, and object caching, including web proxy caches [21] and peer-to-peer media caches [22], which can be deployed to serve the frequent and repeated requests from the cache instead of the original source. However, object-compression and application-layer object-level caching cannot eliminate all the redundant contents alone [1]. Wolman *et al.* [23] estimated that at most 45% of the traffic transmitted from the web servers to the clients to serve the web content requests can be eliminated by using proxy caches based on the Squid [24] cacheability rules. Moreover, neither object compression nor object-level caching work well for contents that have been changed in only minor ways. Delta encoding [25–27], which is based on the differences between versions of a single document, can help in this case by only transferring the changes. However, this also limits the application of delta encoding because both the sender and receiver have to share the same base content, which is not applicable for general Internet traffic. As a software application and network protocol, rsync [28] provides fast files and/or directories synchronization from one location to another, and reduces the bandwidth consumed during file transfers. By default, rsync determines the files to be synchronized by checking the modification time and size of each file, and determines the chunks to be sent by using delta encoding. Thus, rsync lowers the bandwidth consumption during file transfers since only the new files and those parts of the files which have changed will be transferred. Similar to delta encoding, rsync is also not applicable for general redundant Internet traffic elimination.

In order to eliminate the redundancy within individual objects, such as packets, files, or web pages, as well as across objects, protocol-independent redundancy elimination techniques operating on individual packets [1, 3, 9–15] have been explored and investigated recently. Data redundancy elimination (DRE) [8, 29], also known as data de-duplication, is a data reduction technique and a derivative of data com-

Manuscript received 12 Sept. 2012; revised 2nd Feb. and 29 Apr. 2013.

Y. Zhang and N. Ansari are with the Advanced Networking Lab., Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, 07102 USA. (e-mail:yz45@njit.edu and Nirwan.Ansari@njit.edu)

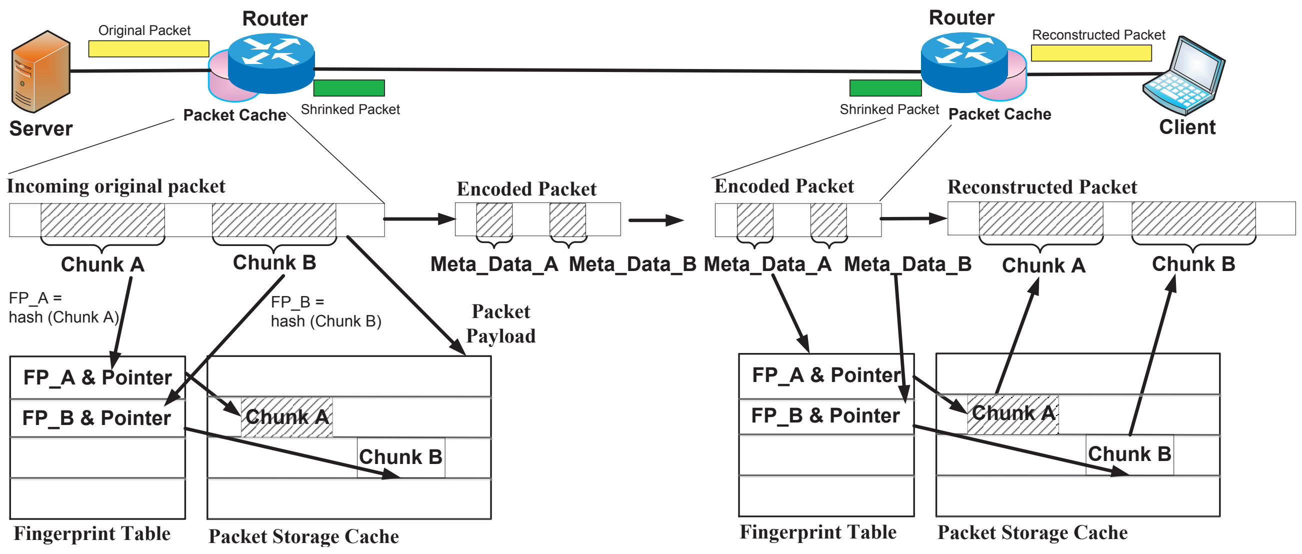


Fig. 1: The classical packet-level data redundancy elimination.

pression. Data compression [20] reduces the file size by eliminating redundant data contained within an object while DRE can identify and eliminate both intra-object and inter-object duplicated data elements, such as an entire file and a data block, to reduce the amount of data to be transferred or stored. The main idea of protocol-independent packet-level redundancy elimination is to identify and eliminate redundant chunks across different packets. When multiple instances of the same data element are detected, only one single copy of the data element is transferred or stored. The redundant data element is replaced with a reference or pointer to the unique data copy.

Protocol-independent redundancy elimination is becoming increasingly popular. Pioneered by Spring *et al.* [1], protocol-independent redundancy elimination techniques have been deployed in wide-area network accelerator middle-boxes, and many vendors, e.g., Cisco [16], Juniper [17], BlueCoat [18], and Riverbed [19], offer such traffic redundancy elimination middle-boxes to improve the network effective bandwidth. Aggarwal *et al.* [3] proposed to deploy DRE at the end hosts to maximize the single-link bandwidth savings, because the amount of contents sent to the destination host can be minimized even for encrypted traffic. Works reported in [9, 10, 30] further expand the benefits of deploying DRE network-wide to eliminate both intra source-destination pair and inter source-destination pairs redundant traffic. Furthermore, great efforts have been explored for redundancy elimination in wireless and mobile environments [31–34]. In addition, PACK (Predictive ACKs) [35], an end-to-end redundancy elimination scheme, has been designed for cloud computing applications.

Based on several tera-bytes of traffic traces collected at different wireline network locations, a large scale trace-driven study on the efficiency of packet-level protocol-independent DRE [2] showed that protocol-independent DRE can achieve average bandwidth savings of 15-60% when deployed at access links of the service providers or between routers. Experimental evaluations on various DRE technologies are presented in [2, 3, 29]. The effects of protocol-independent DRE on redundant traffic elimination in wireless and cellular networks have also been explored by several studies [31, 32, 34, 36]. Based on real-world Wi-Fi and cellular network traces, up to almost 50% bandwidth savings in Wi-Fi networks [31, 36] and as much as 60% mobile data volume reduction in cellular networks [32, 34] can be achieved. All of these studies convinced us that protocol-independent DRE techniques are effective to eliminate redundant traffic over networks. However, it should also be noticed that DRE techniques for identifying and eliminating redundant contents are very expensive in terms of memory and processing capability. In this paper, we provide a survey on the state of the art of protocol-independent data redundancy elimination techniques, including its system architecture and main processes, and some techniques to enhance the DRE performance. We also discuss several DRE systems deployed in wireline, wireless and cellular networks, respectively, in terms of deployment architectures, redundancy identification methods, and cache management.

The rest of the survey is structured as follows. We detail the system architecture of protocol-independent DRE and its major processing mechanisms, including fingerprinting, cache management, chunk matching, and decoding error recovery in Section II. Then, we present several redundancy elimination systems deployed in wireline, wireless and cellular networks in Sections III and IV. In Section V, we discuss several techniques to enhance the DRE performance, such as DRE bypass techniques, non-uniform sampling, and chunk overlap. Finally, Section VI concludes the paper. Unless otherwise stated, DRE techniques discussed in the following sections are restricted to protocol-independent DRE techniques.

II. Components of Protocol-Independent DRE

In this section, we provide a detailed overview on the system architecture and the main processing procedures of protocol-independent DRE techniques. We also discuss several

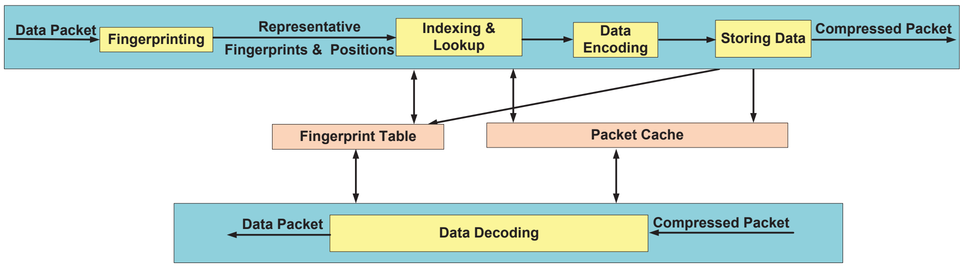


Fig. 2: The main processing blocks of protocol-independent DRE.

major mechanisms activated in protocol-independent DRE, including fingerprinting, cache management, chunk matching, and decoding error recovery.

A. *Overview of Protocol-Independent DRE*

The main idea of protocol-independent DRE is to encode the outgoing packets by replacing identified redundant data chunks with fixed-size meta-data. At the receiver end router, the packets are reconstructed by replacing the encoded content from the packet cache by using the pointer information carried in the encoded packets. The packet-level DRE techniques rely on deploying a fingerprint table and a packet cache at each end of a network path. In general, the DRE algorithms assume that the packet caches located at both ends are always synchronized. Thus, the redundant chunks identified against the packet-cache at the source-side router can be removed from the packets since it can be recovered by the receiver-side router.

A typical protocol-independent packet-level DRE implementation is shown in Fig. 1. For every incoming packet in a particular direction, the DRE algorithm first computes a set of fingerprints by applying a hash function to each chunk of the packet, which is a sub-string of the packet's payload. Limited by the size of the fingerprint table, only a subset of these fingerprints is selected as its representative fingerprints in some way which will be discussed in Section II-B. Both the representative fingerprints and pointers pointing to the locations of the chunks in the packet cache used to calculate its corresponding fingerprints are stored in the fingerprint table. Each representative fingerprint is then checked against the fingerprint table to find a matched data chunk in the packet cache. If such a matched chunk in the packet cache is found, the original data chunk in the packet is encoded with a meta-data, which consists of sufficient information to reconstruct the encoded data chunk at the receiver side, such as fingerprint and the description of the matched chunk range, including a count of redundant bytes before and after the data chunk that is used to calculate the fingerprint. In practice, the size of such meta-data is much smaller than that of the original data chunk. In this example, two chunks are identified as redundant data chunks and the packet is encoded by replacing the original redundant data chunks with the corresponding meta-data. When the other end router receives the encoded packet, it will reconstruct the original packet following the information carried in the meta-data by using the fingerprint table and packet cache at the receiver side.

The main processing stages involved in redundancy elimination include fingerprinting, indexing and lookup, storing data, and data encoding and decoding as shown in Fig. 2. Fingerprinting, also called chunk selection, facilitates the identification of redundant chunks within and across the packets. For every incoming packet in a particular direction, it calculates a set of fingerprints for each packet by applying a hash function to each chunk of the packet and selects a subset of these fingerprints as the representative fingerprints. Each representative fingerprint is then checked against the fingerprint table in the processing stage of indexing and lookup. If one fingerprint already exists in the fingerprint table, a redundant chunk is then identified and its corresponding position of the matched region in the packet cache is also located by using its location information stored in the fingerprint table. Hence, the lookup procedure in this stage involves two parts: fingerprint lookup in the fingerprint table and redundant chunk lookup in the packet store. If one or multiple redundant chunks have been identified in an arriving packet, the packet will go through an encoding procedure by replacing every identified redundant chunk by its corresponding fingerprint description, which consists of the fingerprint as well as the byte range for the matched region in the packet cache. Finally, the new packet is inserted into the packet store and its representative fingerprints are also indexed and stored in the fingerprint table together with the location information of the chunks used to calculate these representative fingerprints in the packet cache. Data decoding performs the reverse operations of data encoding and tries to reconstruct the original packet from the compressed packet by retrieving the chunks from the packet cache by using the information carried in meta-data. The DRE decoder uses the fingerprint value stored in the meta-data to check against the fingerprint table. If such a fingerprint value is found in the fingerprint table, the data chunk will be fetched from the packet cache by using the pointer information stored in the fingerprint table and the count of the redundant bytes before and after the chunk used to calculate the fingerprint. Then, the original packet is reconstructed by replacing the meta-data with these fetched data chunks from the packet cache. If such a fingerprint cannot be found in the fingerprint table at the receiver end router, a decoding error occurs and a recovery process will be activated, which will be discussed in Section II-E. A rather detailed description of the basic operations of DRE can be found in [37].

The main bottleneck limiting the processing throughput of packet-level DRE is the memory access. Assume that the memory can be accessed R times per second at the maximum, and F fingerprints are computed for each packet that has at most k matches. In general, $k \leq F$ because the number of matches can never be more than the number of calculated fingerprints. The packet-level DRE encodings can be applied to at most R/F packets per second because F random memory accesses for each packet are required to check the matched fingerprints and further processing is required to encode the packet if one match is found, while the decoding operations can be applied to at least R/k packets per second because at most k matches can be found for each packet. Thus, $R/F \leq R/k$, implying that the decoding process is much faster than the encoding process. Moreover, suppose that all packets are assumed to be of the same size and the link capacity is assumed to be P packets per second. If the DRE encoding rate R/F packets per second is larger than the link capacity P packets per second, the packets can be DRE encoded up to P packets per second, meaning that line rate DRE encoding and decoding are possible; otherwise, no more than R/F packets can be encoded in every second to ensure line-rate operation, and the decoding rate is also limited by the encoding rate.

As described above, several mechanisms are activated in the implementation of protocol-independent DRE. We will detail these mechanisms in the following sub-sections.

B. Fingerprinting

The goal of protocol-independent DRE is to quickly identify repeated chunks from a stream of packets. So, the first task in protocol-independent DRE is to chunk the packets in an efficient way for redundant content identification. It is impractical to index every fingerprint calculated from each packet in the fingerprint store because nearly every byte in the packet generates one index fingerprint entry. Several chunk selection mechanisms for protocol-independent DRE have been proposed. The main purpose of chunk selection is to generate the representative set of fingerprints for each packet. This process incurs high processing cost and it may also influence the DRE performance. We will review several most popular ones in the following.

1) FIXED: As shown in Fig. 3(a), the FIXED mechanism chunks every p bytes in a packet payload, calculates one fingerprint for each p-byte chunk, and represents the corresponding packet with these calculated fingerprints. By using the FIXED chunking and sampling mechanism, the representative chunks are selected based on position rather than the packet content. It is computationally efficient, achieves exactly the target sampling rate $1/p$, and ensures non-overlapping chunk selection, which means that there are no overlapping among the chunks used to calculate the representative fingerprints for each packet. However, FIXED is not robust against minor modifications in the traffic data; even one word insertion in a text document may fail identifying the redundant chunks. For example, two identical non-overlapping chunks in a document can be identified by the FIXED fingerprinting mechanism. If one word is inserted anywhere between these two chunks, the identification of the redundant chunk can be missed because the redundant chunk could be divided into two different chunks by the FIXED fingerprinting mechanism, thus resulting in the different calculated fingerprints. Hence, the second redundant chunk cannot be identified. Therefore, as shown in [3], this deterministic approach is much less effective than other content-dependent chunking and sampling mechanisms, such as MODP, MAXP and SAMPLEBYTE.

2) MODP: Inspired by a technique developed for finding similar files in a large file system [38, 39], MODP was first applied to the DRE mechanism by Spring *et al.* [1] to remove redundant content chunks across packets. This DRE mechanism was named MODP due to the modular operations and $1/p$ chunk selection rate in the chunking and sampling procedure. The MODP mechanism is depicted in Fig. 3(b) and a formal description of the MODP algorithm is shown in Algorithm 1. For every packet arriving in a particular direction, MODP first computes a set of Rabin fingerprints by applying the Rabin-Karp hash [40] function over sliding windows of w contiguous bytes of the packet's payload (line 5 of Algorithm 1). Given a sequence of bytes $[t_1, t_2, \cdots, t_w]$ of length w bytes, a Rabin fingerprint can be calculated as:

$$RF(t_1, t_2, \cdots, t_w) = (t_1 p^{w-1} + t_2 p^{w-2} + \cdots + t_w) \; mod \; M \tag{1}$$

where p and M are constant integers.

Thus, for an S-byte packet ($S \geq w$), a total of $S - w + 1$ fingerprints can be calculated using the above equation with the substrings $\{[t_1, t_2, \cdots, t_w], [t_2, t_3, \cdots, t_{w+1}], \cdots, [t_{S-w+1}, t_{S-w+2}, \cdots, t_S]\}$. By observing the above equation, the calculation for the next Rabin fingerprint can be simplified by using the previous one as follows:

$$\begin{aligned} RF(t_{i+1}, &t_{i+2}, \cdots, t_{i+w}) = \\ &(RF(t_i, \cdots, t_{i+w-1}) - t_i \times p^w) \times p + t_{i+w} \; mod \; M \end{aligned} \tag{2}$$

Since p and w are constant, $t_i \times p^w$ can be precalculated and stored in a table. With this fast Rabin fingerprint execution, a subtraction, a multiplication, an addition and a modular calculation are required to compute one fingerprint.

It is impossible to store all fingerprints in the fingerprint table, and only those fingerprints whose value is 0 mod p are chosen as the representative fingerprints (line 6 of Algorithm 1) and the fingerprint based on the selected chunk will be stored in the fingerprint table (line 7 of Algorithm 1). Thus, $1/p$ of the calculated fingerprints are sampled as the representative fingerprints for every packet.

Algorithm 1 MODP Mechanism

```
1: //Assume $len > w$;
2: //RabinHash() computes Rabin hash over a $w$ byte window
3: MODP($data$,$len$)
4: for $i = 1; i < len - w + 1; i++$ do
5:     $fingerprint$ = RabinHash($data[i : i + w - 1]$);
6:     if ($fingerprint \; mod \; p == 0$) then
7:         Select data chunk $data[i : i + w - 1]$ and store
           $fingerprint$ in the fingerprint table;
8:     end if
9: end for
```

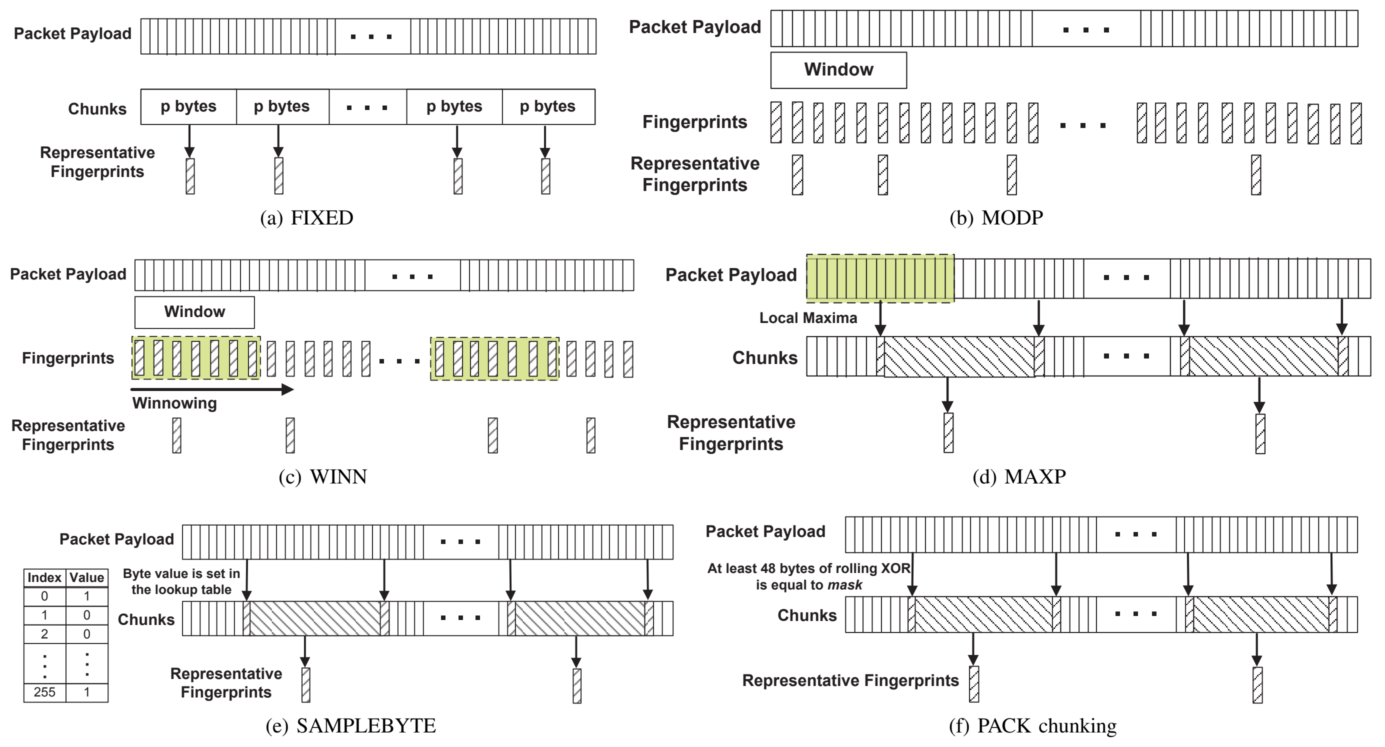


Fig. 3: DRE fingerprinting mechanisms.

Spring and Wetherall [1] investigated the impacts of parameters, sliding window size w and sampling period p, on the redundancy elimination performance and computation overhead. With a large sliding window size w, only large redundant chunks can be identified that might degrade the potential bandwidth savings on small redundant regions. With the decrease of the sliding window size w, smaller redundant regions can be identified, but the computational overhead also increases since more fingerprints need to be calculated. The sampling period p determines how frequent each packet is sampled. A smaller parameter p can increase the probability of finding redundant data chunks, but it also increases the size of the fingerprint index greatly. According to the findings of redundant match distribution [1, 2], small w and p are effective in redundancy elimination, and $w = 32$ or $w = 64$ bytes and $p = 32$ are suggested to maximize the effectiveness of DRE.

As compared to the FIXED mechanism, MODP selects the representative fingerprints for each packet based on the packet content rather than its position, and therefore MODP is robust to small changes of packet payloads: the set of representative fingerprints chosen for a packet remains mostly the same when the packet has been changed slightly. The Rabin-hash computation cost can be reduced greatly when p is a power of 2. The main drawback of the MODP mechanism is the unpredictability of its chunk selection. As reported in [4], the inter-fingerprint distance distribution can have a long tail, indicating that the selected representative fingerprints are not uniformly distributed over the whole packet and large blocks of data remain unselected for fingerprinting. Another shortcoming of the MODP mechanism is that a global parameter, based on which the fingerprints are chosen, needs to be predetermined. In this case, the desired fingerprints sampling rate can be better approximated over a large number of packets, but the fingerprint sampling rate on a per-packet basis could be significantly different from the desired rate.

3) WINN: In order to improve the DRE performance of the MODP mechanism to guarantee a uniform distribution of chosen fingerprints across a data stream, winnowing [41], which was deployed to identify similar documents, has been adapted to identify redundant chunks in Internet traffic. The basic idea behind winnowing in redundancy elimination, as shown in Fig. 3(c), is to track a sliding window over the calculated fingerprint sequence and explicitly choose those fingerprints that are local maxima or minima from the sliding window, and thus winnowing ensures that at least one fingerprint is selected over every specific packet segment. As reported in [4], the distance between two adjacent selected fingerprints in WINN is approximately uniform and always bounded, and the predominating value is equal to the desired fingerprint sampling period. One obvious drawback of WINN over MODP is the required additional computation overhead in searching for local maximum or minimum fingerprints, and thus WINN is slower than MODP.

4) MAXP: As discussed above, a large fraction of calculated fingerprints are not chosen to represent the packet in MODP and WINN, thus incurring high computational overhead required to compute Rabin fingerprints for every packet. A more efficient algorithm, called MAXP [2], has been proposed to improve computation efficiency in MODP and WINN. Fig. 3(d) shows an example of the fingerprinting mechanism of MAXP. MAXP computes fingerprints only when a data chunk has been selected for redundancy elimination.

Algorithm 2 SAMPLEBYTE Mechanism

```
1:  // Assume $len \geq w$;
2:  //LOOKUPTABLE[$i$] maps byte value $i$ to either 0 or 1;
3:  //JenkinsHash() computes hash over a $w$ byte window;
4:  SAMPLEBYTE($data$,$len$)
5:  for $i = 0$; $i < len - w$; $i++$ do
6:    if (LOOKUPTABLE[$data[i]$] $==$ 1) then
7:      Data chunk $data[i : i + w - 1]$ is selected;
8:      $fingerprint$ = JenkinsHash($data[i : i + w - 1]$);
9:      Store $fingerprint$ in the fingerprint table;
10:     $i = i + p/2$;
11:   end if
12: end for
```

MAXP is also based on a local-maximum content-dependent chunking approach which was designed for remote differential compressions [42]. MAXP is quite similar to WINN, but MAXP selects the local-maxima based on the data bytes of chunk instead of among fingerprints as in WINN. Once the local maximum bytes have been identified, the chunk between two adjacent local maxima will be selected and a fingerprint can be calculated based on the selected chunk with an efficient hash function such as Jenkins Hash [43]. Hence, the fingerprint computational overhead is reduced greatly since fingerprints are not required to be calculated first before the local maxima or minima computation as in MODP and WINN. Quite similar to WINN, the selected chunks are distributed approximately uniformly across the data stream [4]. The distance between two adjacent selected chunks is always bounded, and the predominating distance is equal to the desired sampling period. By using the traffic traces from 11 enterprise networks of different sizes and a large university, Anand *et al.* [2] reported that MAXP outperforms MODP by at least 5% and up to 35%. Their results confirmed the advantage of the uniform chunk selection approach on DRE performance.

5) SAMPLEBYTE: Aggarwal *et al.* [3] proposed the SAMPLEBYTE mechanism which selects chunks based on packet content with the computational efficiency of the FIXED mechanism. Fig. 3(e) depicts the SAMPLEBYTE fingerprinting mechanism, and a formal description of the SAMPLEBYTE algorithm is shown in Algorithm 2. The chunk selection of SAMPLEBYTE is based on the most redundant characters in the packet payload. In particular, it utilizes a static 256-entry lookup table with a few specific predefined values, which are generated by an offline, greedy algorithm with the training data. The intuition behind SAMPLEBYTE is that the most redundant characters should have higher opportunities to be selected as chunk boundary markers. The greedy algorithm sorts the characters in descending order of the appearance frequency of the redundant content identified by MAXP. Then, it selects entries of the table to be chunking boundary markers by iteratively choosing the frequently appeared characters, one by one, from the most frequently appeared one until the one which may not yield bandwidth savings.

Similar to FIXED, SAMPLEBYTE scans the packet payload byte-by-byte (line 5 of Algorithm 2). If the data byte value is set in the lookup table (line 6 of Algorithm 2), this byte of data will be identified as a chunk boundary marker. Once a chunk boundary has been identified, a w bytes data chunk will be selected beginning with the identified chunk boundary marker (line 7 of Algorithm 2), and a fingerprint will be calculated using Jenkins Hash [43] based on the selected chunk (line 8 of Algorithm 2). The calculated fingerprint will then be stored in the fingerprint table (line 9 of Algorithm 2). $p/2$ bytes of content are skipped (line 10 of Algorithm 2) after each chunk marker selection to avoid over-sampling when the content data bytes are not uniformly distributed. Since the chunk selection of SAMPLEBYTE is triggered by the positions of those specified pre-determined byte values, the distances between selected chunks are unpredictable. As investigated in [3], SAMPLEBYTE can achieve almost the same bandwidth savings as those of MAXP, but the execution of SAMPLEBYTE is even faster than MAXP because it performs byte-by-byte scanning instead of local maxima searching. Meanwhile, SAMPLEBYTE is robust to small content changes, but one major drawback of SAMPLEBYTE is the static lookup table, and thus an online dynamic adaption of the lookup table is required; this inspires an extension work of SAMPLEBYTE, called DYNABYTE [44, 45].

6) DYNABYTE: In SAMPLEBYTE, the lookup table is static and requires pre-configuration based on the training data. As an extension of SAMPLEBYTE, Halepovic *et al.* [44, 45] proposed an adaptive and self-configuring dynamic algorithm called DYNABYTE (**DYNA**MIC SAMPLE**BYTE**). DYNABYTE follows the same procedures as those of SAMPLEBYTE to remove the redundant chunks. Another problem with SAMPLEBYTE is that the number of entries which is set to "1" in the lookup table is fixed, which may cause over-sampling or under-sampling if it is not set properly. DYNABYTE differs from SAMPLEBYTE in that it utilizes a dynamic and adaptive mechanism to update the lookup table and it also ensures the desired sampling rate. In order to update the lookup table dynamically and achieve the desired sampling rate, two parameters are monitored by DYNABYTE. One is the number of selected chunks, and another one is the byte frequency. DYNABYTE periodically updates the lookup table based on the byte frequencies, and tracks the actual sampling rate. If DYNABYTE is over-sampling or under-sampling more than a pre-defined threshold, two parameters, the number of skipped bytes after one chunk boundary marker has been identified and the number of entries set in the lookup table, can be adjusted to regulate the actual sampling rate to approach the desired value. DYNABYTE always adjusts the number of skipped bytes first. If it cannot be adjusted further, DYNABYTE then switches to adjust the number of entries set in the lookup table.

7) Frequency Based Chunking (FBC): Content-based fingerprinting methods such as MODP and MAXP divide the data stream randomly according to the contents. Frequency based chunking (FBC) [11], an enhancement to the content-based fingerprinting method, explicitly utilizes the chunk frequency information in the data stream to improve the DRE bandwidth saving gains. FBC consists of two processing stages: chunking frequency estimation and chunking. At the first stage,

TABLE I: **Fingerprinting Mechanisms Comparison**

Mechanisms	Chunking Approach	Inter-fingerprint Distance	Computational Overhead
FIXED [3]	Position-based	Uniform distribution	Lowest
MODP [1]	Content-based	Non-uniform distribution with long-tail	High
WINN [2]	Content-based	Approximately uniform distribution and always bounded	High
MAXP [2]	Content-based	Approximately uniform distribution and always bounded	Middle
SAMPLEBYTE [3]	Hybrid of position and content	Unpredictable and the sampling rate is not ensured	Low
DYNABYTE [44, 45]	Hybrid of position and content	Unpredictable, but the desired sampling rate is ensured	Middle-low
FBC [11]	Content-based	Dependent	High
MRC [46]	Content-based	Dependent	High
PACK chunking [35, 47]	Content-based	Non-uniform distribution	Low

FBC incorporates a statistical chunk frequency estimation algorithm, consisting of prefiltering, parallel filtering, and frequency counting to identify the fixed-size chunks with high frequency. Prefiltering and parallel filtering eliminate low frequency chunks as many as possible. In particular, prefiltering applies a modulo-based distinct sampling while parallel filtering utilizes parallel bloom filters to identify high-frequency chunk candidates. Only those highly-duplicated chunks can survive from these two filtering steps, and finally their frequencies are estimated at the frequency counting step.

8) Multi-Resolution Chunking: Chunk size can directly affect the bandwidth saving gains, storage performance and memory pressure. Small chunks may achieve better bandwidth savings at the cost of increasing the total number of chunks, and consequently increase the fingerprint index entries, and increase the memory pressure and storage seeks. Large chunks can reduce the memory pressure and provide better storage performance since more data can be read through one read, but bandwidth saving gains might be degraded with large chunks since fine-grained changes can be missed. To combine the advantages of fingerprinting with large and small chunks, Multi-Resolution Chunking (MRC) [46] has been proposed to allow multiple chunk sizes to co-exist in the redundancy elimination system. MRC can achieve a high content compression rate with low memory overhead and low storage seeks. MRC uses larger chunk size to reduce the storage seeks and memory pressure for large matching region, and uses smaller chunk size to achieve higher bandwidth savings when relative small redundant regions are missed with large chunks.

In order to avoid the chunk boundary overlap caused by different chunk size fingerprinting, a MRC tree is constructed with the largest chunk as the root and smaller chunks as the leaves. All of the smaller chunks share boundaries with some of their leaf chunks. The MRC tree is generated by detecting all of the smallest boundaries first, and then matching up different numbers of bits to generate larger chunks with the same boundary detection constraint. With this single-pass fingerprinting process, a cleaner chunk alignment can be generated with less computational overhead.

9) PACK Chunking: Based on an XOR rolling function, PACK chunking [35, 47] is a computationally efficient fingerprinting scheme. The PACK chunking algorithm is presented in Algorithm 3. Its basic idea is that a chunk boundary marker is found if the rolling bitwise XOR of at least 48 bytes is equal to a predefined value (lines 6 and 7 of Algorithm 3). Then, the representative fingerprints are calculated based on the selected chunks with SHA-1. Fig. 3(f) shows the operations of PACK chunking. The measurements in [35, 47] showed that the execution speed of PACK chunking is even faster than SAMPLEBYTE with 8 entries in the lookup table set.

10) Comparison: The fingerprinting mechanism plays a critical role in the performance of DRE, and at the same time it also incurs high processing costs. A comparison among all of the discussed fingerprinting mechanisms above is summarized in Table I in terms of the chunking approach, inter-fingerprint distance, and computational overhead. FIXED is a simple content-agnostic and position-based chunking mechanism. The representative chunks are selected uniformly across the data stream. However, any byte insertion or deletion in the data may change all the data chunks, resulting in the failure of redundant chunk identification and elimination. The chunk boundary identification in FIXED does not involve any computation. Thus, FIXED incurs the lowest computational overhead among all of these fingerprinting mechanisms.

The chunking approaches in MODP, MAXP and WINN are all content-based. Hence, they are robust to small data changes. The inter-fingerprint distance is not uniformly distributed and has a long tail, indicating that some large blocks of data remain unselected for fingerprinting. WINN and MAXP improve the chunk selections by ensuring one chunk is selected over one local region. The inter-fingerprint distance distribution is

Algorithm 3 PACK chunking algorithm

```
1: $mask \leftarrow 0x00008A3110583080$ {48 bytes window; 8 KB chunks}
2: $longval \leftarrow 0x0$ {$longval$ 64 bits length}
3: for all ($byte \in stream$) do
4:     shift left $longval$ by 1 bit {$lsb \leftarrow 0$; drop $msb$}
5:     $longval \leftarrow longval$ bitwise-xor $byte$
6:     if (processed at least 48 bytes and $longval$ bitwise-and $byte == mask$) then
7:         found an anchor;
8:     end if
9: end for
```

approximately uniform to the desired sampling period and is always bounded. Owing to the large amount of Rabin hashes calculations, MODP leads to a high computational overhead, while WINN deteriorates the conditions by local maximum or minimum fingerprint searching. MAXP simplifies the calculations by local byte value maximum searching.

SAMPLEBYTE and DYNABYTE capitalize on the robustness of a content-based approach with the computational efficiency of a position-based approach. The inter-fingerprint distances are unpredictable with both mechanisms. The desired sample rate is not ensured in SAMPLEBYTE, but is improved in DYNABYTE. The computational overhead is further reduced by byte scanning and comparison instead of local maximum byte searching in MAXP. DYNABYTE induces higher computational overhead than SAMPLEBYTE in dynamic updates of the lookup table and the adjustment of the actual sampling rate.

FBC, MRC and PACK chunking are all content-based fingerprinting algorithms. Both FBC and MRC are enhancements based on another content-based fingerprinting algorithm, and therefore the inter-fingerprint distances are also dependent on that particular fingerprinting algorithm. FBC and MRC incur additional computational overhead to estimate chunk frequencies and to generate MRC trees, respectively. A large number of hash operations are especially incurred in the stage parallel filtering of the FBC algorithm. The inter-fingerprint distance with PACK chunking is not uniformly distributed. For each byte in the data stream, only one bitwise XOR operation is applied in PACK chunking, while multiple bitwise XOR operations are required for SAMPLEBYTE fingerprinting to compare the byte value with the entries set in the lookup table. Therefore, more computational resources are required for the PACK chunking algorithm.

C. Chunk Matching

In general, two chunk matching mechanisms can be used for protocol-independent DRE techniques [3]. One is called "Chunk-Match" and the other one is "Max-Match". If one matched representative fingerprint is found in the fingerprint table, the w byte representative region used to compute this representative fingerprint is identified as the redundant chunk by the Chunk-Match. While in Max-Match, the matched region is maximized by expanding to the left and right of the representative region. Chunk-Match may miss some similar redundant regions. Max-Match can overcome this problem, but it suffers from excessive memory access induced by byte-by-byte comparison.

In order to reduce the number of memory access and at the same time maximize the matching region, CombiHeader [12] uses a chunk aggregation technique to expand the matching chunks and reduce the DRE encoding overhead penalties. CombiHeader is based on the assumption of the spatial locality property of popular chunks, i.e., consecutive popular chunks have high probability of appearing together. In addition to track the hit counts of each chunk, CombiHeader also maintains the link-hit counts between two adjacent chunk pair. Based on the link hit counts between two adjacent chunks, these two chunks may be merged into a new one if the link hit counts exceed some threshold. Then, if the first chunk is matched, CombiHeader will check the next chunk if it can be matched or not; if so, CombiHeader encodes them together instead of two separated chunks. In this way, the matching region can be expanded and the encoding overhead penalty can be reduced.

D. Cache Management

The packet store used to cache previously observed packets for redundancy elimination is limited in size. Usually, First-In-First-Out (FIFO) cache replacement policy is assumed. Two bandwidth savings based cache replacement policies, namely, Least Recently Used (LRU) and Least Savings with Aging (LSA), have been proposed in [4]. LRU is inspired by the observation that popular chunks exhibit temporal locality [2, 4]. The least recently used chunks will be removed from the cache with the LRU cache replacement policy when space is needed. As compared to FIFO, 5% more relative bandwidth can be saved with LRU on average, but the execution time also increases by up to 40%.

LSA is an extended policy of Least Frequently Used (LFU) with aging. LSA considers the properties of popular chunks [2]: popular chunks exhibit temporal locality, and the popularity of chunks exhibits a Zipf-like power-law distribution. Instead of using the cache hits, LSA uses the actual byte volume contributed to bandwidth savings to rank the chunks, and then LSA removes the chunk with the least bandwidth savings when space is needed. Hence, the data chunks which are more popular than others are kept in the cache for a longer time. One problem with LRU is that some chunks may stay too long in the cache and never get replaced. Considering the temporal locality property of the popular chunks, LSA purges the entire cache periodically as a form of aging to avoid cache pollution by some data chunks. As reported in [4], LSA performs better than FIFO and LRU. LSA consistently produces higher bandwidth savings than FIFO does. In addition, the execution time of LSA is comparable to that of FIFO.

E. DRE Decoding Error Recovery

Packets could be lost due to congestions or transmission errors, which may prevent DRE encoded packets from reconstructing the original data packets. This is called DRE decoding error recovery. Most of DRE techniques for wired networks retransmit the lost packets [1] to recover the DRE decoding errors. Several DRE decoding error recovery schemes, such as ACK Snooping [34], Informed Marking [34], and Post-ACK caching [36], have been proposed to improve the DRE performance in a high packet loss environment such as WLAN and cellular networks.

1) Retransmission-based scheme: With the retransmission-based decoding error recovery scheme, the receiver just simply requests the retransmission of the missing packet from the sender when a cache missing is hit. In a packet-loss environment, the DRE with the retransmission-based decoding error recovery scheme can achieve almost the same bandwidth

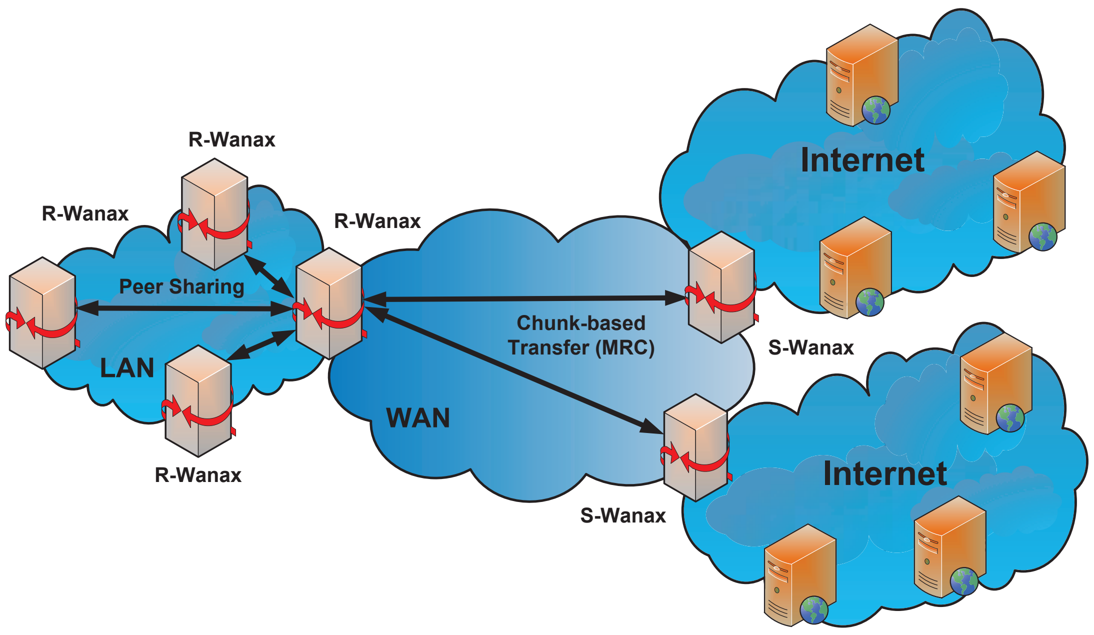


Fig. 4: Wanax redundancy elimination system overview [46].

savings as that with no packet losses, but the retransmission-based decoding error recovery scheme could waste significant additional bandwidth, especially in a high-loss environment.

2) ACK Snooping: ACK Snooping [34] based decoding error recovery scheme relies on the feedback acknowledgements of the correctly received packets of the transport protocols, such as TCP. With ACK Snooping, the sender needs to maintain a blacklist of packets that are not acknowledged for some predetermined time period after they are transmitted. ACK Snooping can quickly detect the lost packets and reduce the DRE cache missing rate to 0 without introducing additional feedback overhead. However, lost packets could be false negatively estimated with delayed acknowledgements in ACK Snooping, and it cannot work with transport protocols without a feedback acknowledgement mechanism such as UDP, which is common in wireless networks.

3) Informed Marking: The basic idea of Informed Marking based decoding error recovery scheme [34] is that the receiver informs the sender with the hash of the missing packet, and the sender puts the corresponding packet into the blacklist of the missing packets in the receiver's cache and ignores the redundant chunk matched to any blacklisted packet from future DRE encodings. Informed Marking can reduce the DRE decoding error rate to at most the network packet loss rate since the lost packet can only have at most one time opportunity to evoke the DRE decoding errors. Unlike ACK Snooping, Informed Marking is more general and can work with any type of transport protocols such as TCP and UDP, and it does not introduce any unnecessary feedback overhead from the receivers when there is no packet loss or when the lost packets are not used in DRE encoding.

According to the study in [34], all of these three DRE decoding error recovery schemes, namely, Retransmission-based scheme, ACK Snooping, and Informed Marking, are able to remove DRE decoding errors. Among these three decoding error recovery schemes, Informed Marking is more general and does not introduce any unnecessary feedback overhead. As the study results shown in [34], more than 60% of the bandwidth savings by applying DRE can be preserved by the Informed Marking decoding error recovery scheme, even in cases with high packet loss rates.

4) Post-ACK Caching: Post-ACK Caching [36], a MAC-layer ACK based decoding error recovery scheme, has been proposed for WLAN. In WLAN, a MAC-layer ACK will feedback to the sender for every correctly received frame. In the Post-ACK Caching scheme, all the MAC-layer frames containing IP packets whose payloads are used for redundancy elimination must be acknowledged by the receiver, in order to ensure that all the packets used for redundancy elimination have been received properly at the receiver. As reported in [36], POST-ACK Caching can achieve more than 20% better bandwidth savings in WLAN than black-listing based decoding error recovery schemes, such as ACK Snooping and Informed Marking, can.

III. DRE SYSTEMS FOR WIRED NETWORKS

A typical redundancy elimination middlebox implementation has been introduced in Section II-A. Several redundancy elimination systems have been proposed. In this section, we will discuss the implementations of several redundancy elimination systems.

A. *Wanax*

Wanax [46] is a wide-area network acceleration middlebox, designed for developing world deployments where storage and WAN bandwidth are scarce. Three mechanisms are activated in Wanax, namely, multi-resolution chunking (MRC), intelligent load shedding (ILS) and a mesh peering protocol, to achieve high compression rate and high storage performance with small memory pressure, to maximize effective bandwidth by adjusting storage and WAN bandwidth usage, and to reduce latency by fetching content from relatively higher-speed local peers instead of over slow WAN links when the content is available from other local peers, respectively. Wanax works by removing redundant traffic between a pair of WAN acceleration middlebox, called S-Wanax and R-Wanax, which are located near the source and destination, respectively.

Fig. 4 shows the system architecture of Wanax. S-Wanax holds back from sending the original TCP stream to R-Wanax, and instead sends the missed chunk signatures at R-Wanax from the MRC tree, which is generated by the MRC process. In order not to waste the bandwidth by sending the full MRC trees to R-Wanax, S-Wanax maintains a hint table, which contains recently-seen chunk names along with timestamps, and prunes the MRC tree to avoid sending the sub-trees below any un-stale chunk whose name is listed in the hint table. By receiving the chunk signatures from S-Wanax, R-Wanax incorporates a peering mechanism to fetch the missed chunks. It checks chunks listed in the message from S-Wanax in its local cache first. If a missing chunk hits, it sends a chunk request message, including the missing chunk name and the address of S-Wanax, to its responsible peer which is determined by a variant of consistent hashing called Highest Random Weight (HRW) [48] for a particular chunk. If the missing chunk is found at the peer R-Wanax, it responds with a chunk response message. Thus, incorporating with the peering mechanism, R-Wanax can distribute the chunk fetching load among the peers and utilize multiple chunk caches in parallel to improve the effective bandwidth performance by deploying the ILS mechanism, which exploits the structure of the MRC tree and dynamically schedules chunk fetches based on the estimated fetch latencies of network and storage queues. Naturally, Wanax incurs a three-way handshake latency for non-cached data.

B. Network-wide Redundancy Elimination

The typical DRE middlebox implemented in commercial WAN accelerators can eliminate redundancy over a single end-to-end network link. Deploying redundancy elimination on a single link can only remove intra source-destination pair redundant traffic and cannot remove inter source-destination pairs redundancy. Network-wide redundancy elimination [9, 10, 30] by traversing potentially duplicate packets onto common links can remove both kinds of redundancy.

Network-wide redundancy elimination was originally proposed by Anand *et al.* [9]. Given the physical network topology and traffic redundancy profiles, each of which consists of a sequence of packets to be transmitted from a source to a destination and the redundancy among all of these packets, a linear optimization problem is formulated to find an overlay network which routes the traffic among all the source-destination pairs with the minimum cost of total network resources when the redundancy elimination is enabled at all routers. The proposed linear problem subjects to link capacity

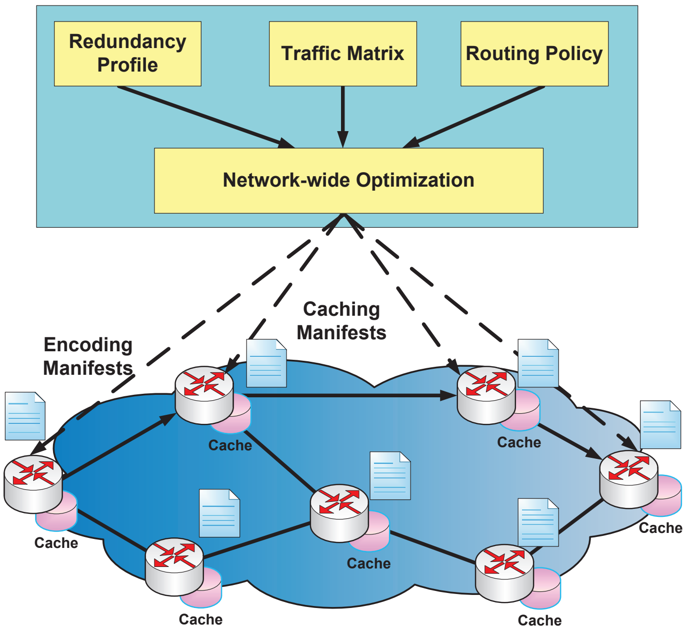


Fig. 5: Schematic depiction of SmartRE [10].

and flow conservation constraints. As an extension to the work proposed in [9], a greedy heuristic algorithm for redundancy-aware routing with limited resources is proposed in [30]. The proposed greedy algorithm iteratively reroutes the traffic for each source-destination pair to maximize the network resource cost reduction until the number of overlay nodes reaches its maximum limit, or the traffic for all the source-destination pairs has already been rerouted, or the network cost cannot be reduced further. The proposed network-wide redundancy-aware routing algorithm in [9] is somewhat similar to multicast routing algorithms [49], with which content is delivered from the source to a group of destinations simultaneously in a single transmission. The main difference of the network-wide redundancy-aware routing from multicast routing is that the content distribution routing paths with the redundancy-aware routing algorithm are influenced by destinations which have observed significant redundant contents, while the multicast routing algorithms consider simply the multicast participants.

Although network routes have been optimized to maximize the redundant packets traversing through the common links, the network-wide redundancy elimination in [9, 30] still operates in a hop-by-hop fashion. The redundancy elimination on different routers along a network path works independently. A coordinated network-wide redundancy elimination [10], called SmartRE, has been proposed to leverage the memory resources efficiently across multiple router hops in a path to improve the DRE performance. As shown in Fig. 5, SmartRE redundancy elimination consists of three key elements: ingress nodes, interior nodes, and a central configuration module. The ingress nodes and the interior nodes only store a subset of packets they observe as instructed in the encoding manifests and caching manifests generated by the network-wide optimization module. A caching manifest specifies the caching responsibility of each interior node in terms of a hash-range per path per node, while an encoding manifest specifies the total covered hash range of the path. SmartRE incorporates ideas from cSamp [50] to split caching responsibilities across multiple routers over a network path. For each path, the central configuration module generates a set of caching manifests by solving a linear programming problem to optimize the network operator's objectives by using network-wide redundancy profiles, traffic matrix, routing policies, and resource constraints. For each path, the ingress node encodes packets and stores the packets whose hashes fall in the total covered range of the path, while the interior node decodes the packets and only stores the packets whose hashes fall within the range assigned to it for the path. SmartRE enables the coordination of the available resources at network routers, and thus can improve the resource effective utilization and magnify the overall network-wide DRE performance.

C. PACK

Measurements [35, 47] have shown that end-to-end DRE solutions incur considerable computational and storage cost on servers, which may eradicate the bandwidth savings achieved by the DRE in cloud computing systems. Thus, DRE techniques deployed for cloud computing are required to optimize the bandwidth saving gains with the additional cost of computations and storage incurred by DRE. In order to relieve the cloud's computational and storage cost induced by DRE, Zohar *et al.* [35, 47] proposed a receiver-based end-to-end DRE system, namely, PACK, for cloud computing. Unlike the classical DRE technique, PACK off-loads the cloud-server DRE computational effort from the cloud to a large group of end-clients by using the power of prediction at the receiver side to remove redundant traffic.

Fig. 6 illustrates the operations of redundancy elimination procedures in PACK. During the initial TCP handshake, the sender and the receiver can negotiate to enable the PACK redundancy elimination by adding a "PACK permitted" flag in the TCP's options field. Then, the sender starts to send the data in TCP segments. The receiver parses the received data stream to a sequence of variable sized chunks using the "PACK chunking" scheme discussed in Subsection II-B9, computes the respective signature for each chunk using SHA-1, and adds the parsed chunks and their associated meta-data to the local chunk store in a "chain" scheme. The chunk's meta-data includes the chunk's signature and a single pointer to the successive chunk. With the newly parsed chunks' signatures, the receiver looks for a matched signature in its local chunk store. If a matched signature is found, the receiver retrieves a sequence of chunks' signatures using the chunks' pointers

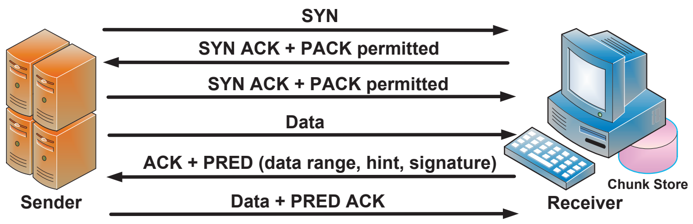


Fig. 6: PACK redundancy elimination system [35, 47].

included in the chunk's meta-data and sends a prediction, including the range of the predicted data, the hint (e.g., the last byte in the predicted data, or the checksum of all or part of the predicted data) and the signature of the chunk, to the sender for the subsequent data with a "PRED" message in the TCP's option filed. Upon receiving a "PRED" message from the receiver, the sender determines the TCP sequence range and verifies the hint for each prediction. If a hint matches, the sender calculates and verifies the chunk signature for the predicted data range; otherwise, this prediction fails. If the signature is also matched, the sender replaces the corresponding outgoing buffered data with a "PRED-ACK" message to remove the redundant data.

The drawbacks of PACK are also rather obvious. First, redundancy elimination is limited to TCP traffic only. Moreover, additional traffic load is introduced by the "PRED" messages from the receiver to the sender. Also, bandwidth saving gains are only partial since there are always delays between the redundancy detection at the receiver and redundancy elimination performed at the sender. This delay is more than 100 ms in general because cloud applications are deployed over a wide-area network in current days.

D. Hybrid Chunk- and Object-level DRE

Traffic redundancy elimination can work on different granularity levels: object level or chunk level. Object-level cache provides less bandwidth saving gains than chunk-level redundancy elimination. However, chunk-level DRE techniques have limitations in computational speed, memory and storage overhead. Chunk-level DRE computational overhead may become a bottleneck [52], especially in higher bandwidth links. A hybrid chunk and object level redundancy elimination technique has been proposed for web optimization in [51]. Fig. 7 shows the architecture of the proposed hybrid DRE system. Two middleboxes, the encoding middlebox close to the source and the decoding middlebox near the destination, work cooperatively over a wide-area network (WAN) to remove the redundant traffic. The encoding middlebox consists of a scheduler, a chunk-level DRE module, an object-level proxy cache module, a RAM memory module, a persistent storage for chunk and object caching, and a bypass module. The chunk-level DRE module and the object-level proxy cache module share the memory and the persistent storage.

The functions for the main modules are summarized as follows:

1) Scheduler: The scheduler determines which module the data stream should flow through, the DRE module, the proxy cache, or the bypass module. The scheduler tracks the traffic with destination port 80, processes the HTTP headers and classifies the applications to determine whether the object is cacheable or not based on RFC 2616 [53]. All cacheable objects will be flowed toward the proxy cache module and all non-cacheable contents will be flowed through the DRE module. The scheduler may direct the remaining TCP and UDP traffic to the DRE module or the bypass module.

2) Proxy Cache Module: For a cacheable object, the proxy cache module hashes its URL and stores the hash value in the

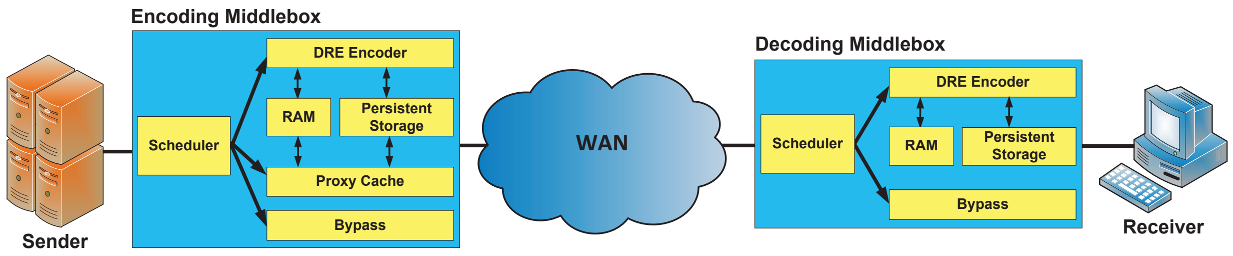


Fig. 7: Hybrid cache and DRE system [51].

RAM together with some other meta-data of the object. The object is then stored in the persistent storage. When an object is requested and is still valid in the proxy cache, it will be retrieved from the cache, otherwise it will be fetched from the source server.

3) DRE Module: The chunk-level redundancy elimination is performed in the DRE module. All the selected chunks are stored in the persistent storage while their corresponding fingerprints generated in the DRE calculations and their pointers point to the cache location where the actual content is stored in the RAM.

With this hybrid architecture, the computational and memory overhead can be lowered because part of the data stream is directed through the proxy cache module without hash calculations over chunks and the corresponding representative fingerprints storage in memory.

IV. DRE Systems for Wireless and Cellular Networks

A majority of protocol-independent redundancy elimination techniques focus exclusively on fixed wireline networks. Considering the high error rate of wireless channels and bandwidth resource scarcity of radio spectrum, DRE emerges as a promising way to save bandwidth and improve performance in wireless environments. Leveraging redundancy elimination in wireless and mobile environments has been explored in several recent works [31–33]. In this section, we will first explore the network traffic redundancy in wireless and cellular networks. Then, we will discuss the challenges and opportunities to leverage redundancy elimination in wireless and cellular networks. Finally, we will detail the redundancy elimination techniques proposed especially for wireless and cellular networks.

A. Traffic Redundancy in Wireless and Cellular Networks

Previous works have shown that a considerably large amount of redundant contents exist in wireline networks at ISP access links, between routers, or between end-to-end hosts. The traffic redundancy in wireless and cellular networks has also been explored by several recent studies [31, 32, 34, 36]. Depending on the nature of redundancy, two types of redundancy have been observed: intra-user temporal redundancy and inter-user redundancy.

Based on real network traces of two Wi-Fi networks, Zhuang and Sivakumar [31] explored both types of redundancies. Up to 75% redundant traffic can be observed by investigating one particular day's traffic of one individual user against the previous day's traffic of the same individual user. They also showed that eliminating inter-user redundancy in Wi-Fi networks can improve bandwidth savings ranging between 7% to 22%. Another redundancy analysis [36] of wireless user traffic from a campus WLAN showed a wide range of redundancy across traces: for some traces, up to almost 50% bandwidth saving can be achieved, but quite low redundancy as small as around 2% could also be observed for some traces, which may simply waste resources in processing and encoding overhead of DRE.

Using three large real-world cellular network traces, Lumezanu *et al.* [34] reported that mobile users may save as much as 60% traffic volume by deploying redundancy elimination. An average of 46% intra-user individual temporal redundancy in cellular web traffic has been estimated conservatively by Zohar *et al.* [32] by using a 5-hour traffic trace of a cellular gateway which connects 130 web sites to the cellular networks. From the above studies, we can see that a significant amount of redundant traffic exists in both wireless and cellular networks.

B. Challenges and Opportunities of DRE in Wireless and Cellular Networks

Given the error-prone nature of wireless channels and bandwidth resource scarcity of radio spectrum, applying DRE to wireless and cellular networks is promising in saving bandwidth and improving network performance. However, unlike wire networks, wireless and mobile environments exhibit unique challenges and opportunities in the context of redundancy elimination. The broadcast nature of wireless communication makes it easier to eliminate inter-user redundancy by traffic overhearing. On the other hand, high packet loss rate, user mobility and some MAC layer characteristics could impose severe challenges to DRE efficiency in the wireless environment.

Halepovic *et al.* [36] explored the effectiveness of DRE in wireless local area networks (WLANs) and they also explored specific issues affecting DRE performance in WLAN. Their results indicate that the effectiveness of DRE is lower than that of Ethernet due to higher physical channel error rates in WLANs than in Ethernet, smaller proportion of IP traffic, and MAC-layer characteristics, such as longer MAC headers, control and management frames, retransmission, and frame drops.

Packet loss in the wireless environment due to the transmission error over the air interface, insufficient buffer sizes or communication congestion are normal in wireless environments. The high packet loss rate makes it difficult to apply DRE in wireless and cellular networks. The effects of packet loss on redundancy elimination, in terms of bandwidth savings and DRE decoding errors, in cellular wireless networks have been explored in [34]. They reported that DRE could be disrupted by losing only a few packets. Packet loss may cause DRE decoding errors, which will prevent receivers from reconstructing the original packets.

C. DRE Algorithms for Wireless and Cellular Networks

In this section, we will review most of DRE algorithms deployed specially for wireless and cellular networks.

1) Wireless Memory AP-client DRE: Wireless Memory (WM) [31], a two-ended access-point(AP)-client solution, was proposed to explore traffic redundancy elimination in wireless and mobile environments, especially in Wi-Fi (802.11b/g) networks. Both Intra-user and inter-user traffic redundancy can be removed by WM. As shown in Fig. 8, WM maintains memory space in both AP and clients, and AP needs to maintain separate memory for each of its clients when it communicates with multiple clients simultaneously. Certain WM-related information, e.g., memory space size and delimitation parameters, needs to be initialized when a WM-enabled client is first associated to a WM-enabled AP. WM is designed to be application transparent and works at the packet level residing between the network layer and the link layer, and thus no application needs to be changed. WM has two basic complementary operations: Memory Referencing and Memory De-referencing (MRD). Built with three sequentially invoked components of Delimitation, Memory Lookup, and Packet Composition, Memory Referencing eliminates the redundant contents from the original data packet against the memory and reconstructs the packet by replacing the redundant data segments present in the memory with a code which represents the memory entry, while Memory De-Referencing recovers the original data packet through three complementary sequential components of Packet De-composition, Memory Lookup, and Packet Assembly.

WM can effectively remove intra-user redundancy through MRD, and it can be enhanced with Memory Fidelity Enhancement (MFE) to eliminate inter-user redundancy by overhearing other clients' traffic. MFE requires clients to actively overhear other clients' traffic whenever possible, and the overheard data will be put into its memory for redundancy elimination. In order to encode the traffic to one particular user with other users' traffic, AP needs to estimate which client's traffic has been overheard and decoded by this particular user based on the fact that clients can always overhear and decode the traffic that are sent with lower rates. As an example scenario shown in Fig. 8, client C_j can overhear and decode the traffic of client C_i while client C_k cannot since the data rate of C_i is lower than that of client C_j and larger than that of client C_k. Hence, the redundant segment D to client C_j can be replaced by its reference d to the previous segment of client C_i while

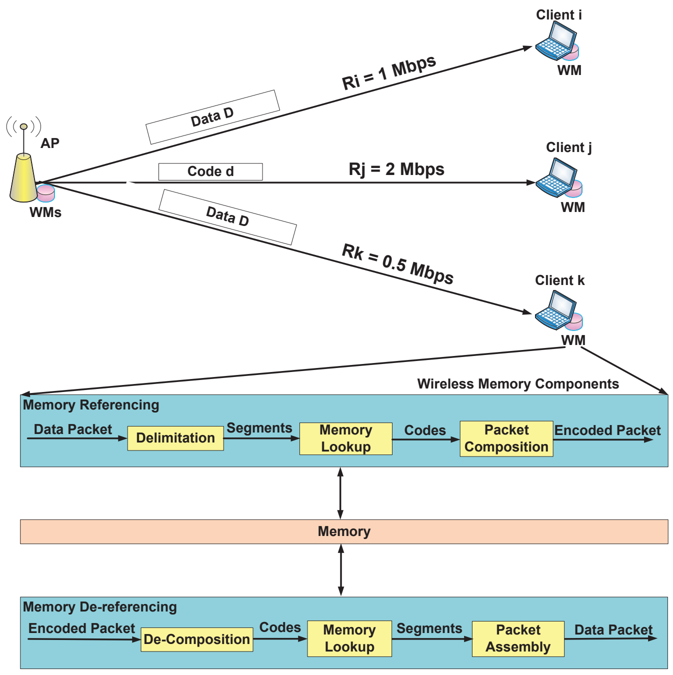


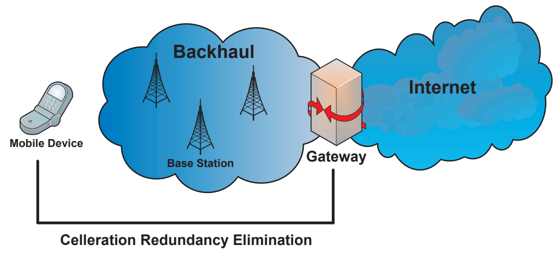
Fig. 8: Illustration of Wireless Memory [31].

client C_k cannot take advantage of this since its data rate is lower than that of client C_i.

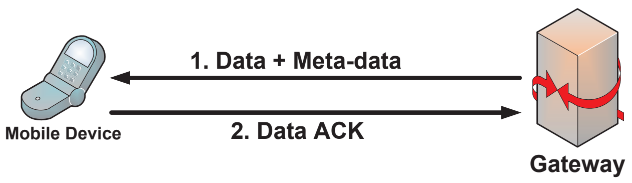
Overall, the operations of MRD and MFE complement each other to improve DRE performance by removing both intra-user and inter-user redundancies. When WM receives an original data packet, MRD will replace the redundant segment with a reference to a memory entry; when an encoded packet is received, MRD extracts the codes and reconstructs them back to the original packet. MFE requires clients to actively overhear other clients' traffic and put the overheard traffic into its own memory for inter-user redundancy elimination.

2) Celleration gateway-to-mobile DRE: Celleration [32], a loss-resilient gateway-to-mobile traffic redundancy elimination system, was designed for data-intensive cellular networks as shown in Fig. 9(a). It can remove a considerable amount of redundant traffic at both the back-haul and the wireless last-mile of the network while preserving handset battery power. Celleration activates three mechanisms at the gateway, namely, flow coding, ad-hoc learning of mobile devices, and flow reduction, as illustrated in Fig. 9(b-d), respectively. A Celleration-enabled gateway, located at the cellular network Internet entry point, extracts similarities in repetitive chunk flows across users with the flow coding mechanism, predicts individual mobile user's future data with ad-hoc learning mechanism, and enables bandwidth savings for mobile end-users with the flow reduction mechanism.

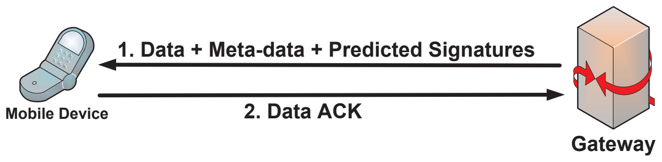
In the flow coding mechanism as outlined in Fig. 9(b), the gateway enabled with the flow coding mechanism continuously parses the crossing flow to a sequence of variable sized, content-based chunks, which will be signed by using SHA-1. The chunks' signature sequences will be stored in the gateway's local cross-user signature store. At the same time, these



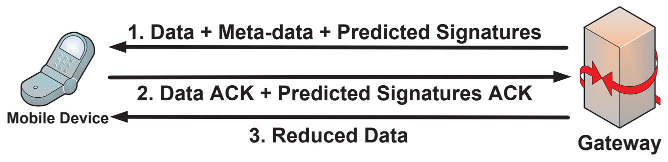
(a) Celleration's Deployment Architecture.



(b) Flow Coding.



(c) Ad-hoc learning.


1. Data + Meta-data + Predicted Signatures
2. Data ACK + Predicted Signatures ACK
3. Reduced Data
Mobile Device
Gateway

(d) Flow reduction.

Fig. 9: Celleration redundancy elimination system.

chunks' signature sequences together with their corresponding chunks' size information are sent as meta-data to the mobile device to save the computation overhead involved in DRE chunking and signing. Celleration can recognize a crossing flow by its chunks' signatures, which will also be used to look up the potential future data in the cross-user signature store. Once a crossing flow has been recognized, the ad hoc learning between the gateway and the mobile device, the operations as described in Fig. 9(c) will be activated. The gateway sends a list of predicted future chunks' signatures to the mobile end user by using the previous flows of all users stored in the cross-user signature store. The mobile end device will respond with a list of time-limited approvals by checking whether the corresponding data of the predicted chunks' signatures are in its local cache or not. If the ad-hoc learning indicates that some predicted data chunks already exist in the cache of the mobile device, the flow reduction mechanism of Celleration as illustrated in Fig. 9(d) will be activated by refraining the gateway from forwarding the approved chunks to the mobile device as they arrive at the gateway. Thus, the redundant traffic over the transmission path between the gateway and the mobile device can be removed.

3) REfactor: Previous efforts on leveraging the broadcast nature of the wireless environment in redundancy elimination focused on overhearing entire packets, which may limit the benefits of fully leveraging opportunistic overhearing. By deploying opportunistic overhearing over sub-packet level packet content, REfactfor [33] can improve the IP-layer redundancy elimination performance. REfactor uses the MAXP algorithm with Chunk-Match for data fingerprinting. As an improvement, it incorporates a self-addressing mechanism wherein a chunk can identify its storage location in the cache based on the chunk content, and a removed chunk can be identified by its location in the cache. Thus, the encoder can only maintain the chunk hashes in a hash table and the decoder can only maintains the chunks in a FIFO cache. Based on the fact that overhearing probabilities are time-varying and are different for different clients, REfactor introduces a reception probability vector for each chunk entry stored in the hash table at the AP. By using this reception probability vector, REfactor estimates the expected benefit, which is measured as the reduction in transmission time by eliminating a redundant chunk. If the expected benefit exceeds some predefined threshold, the redundant chunk will be removed. The retransmission-based decoding error recovery scheme is deployed in REfactor. If a cache miss hit occurs at the receiver, the receiver requests a chunk retransmission from the sender.

V. Other Techniques to Enhance Redundancy Elimination Performance

Several techniques [4, 37, 54–56], such as DRE bypass techniques and content-aware chunk selection, have been proposed to enhance the DRE performance in terms of computational overhead, execution time and bandwidth savings. In this section, we investigate the effects of these techniques on the performance of DRE.

A. Bypass Techniques

The goal of DRE bypass techniques is to save the resource overhead including computation, memory and storage, induced by DRE operations for processing the traffic which has no or little contribution to redundancy elimination. Several DRE bypass techniques [4, 51, 55, 57] based on the packet size, port number, or traffic type have been proposed to enhance redundancy elimination performance.

Motivated by the observation of bimodal distribution of Internet packet size [58], a size-based bypass technique [4] has been investigated to improve the DRE performance in terms of execution time and bandwidth savings. Execution time can be reduced since fewer packets will be processed by the sized-based bypass enabled DRE technique. However, bandwidth saving gains might be degraded with improperly set bypass packet-size since too many data are unprocessed for redundancy elimination. Therefore, a threshold needs to be setup for size-based bypass technique. 96 bytes in transport packet payload are suggested for size-based bypass threshold, and this setting seems to offer significant improvement in execution time and slightly higher bandwidth savings.

Secure web traffic (port 443) should not be processed for redundancy elimination since the packet payload is encrypted and shows little redundancy [1, 2]. Actually, the hybrid chunk- and object-level DRE middlebox [51] discussed in section III-D also incorporates a kind of DRE bypass technique based on port number and content type. With a hybrid chunk- and object-level DRE middlebox, only the un-cacheable contents over TCP transport protocol will be directed to the DRE module, while the cacheable contents over TCP are offloaded to the proxy cache module, and the remaining TCP and UDP traffic could be bypassed without any processing.

As reported in [59, 60], typical binary files such as video, audio, binary executable, and compressed files match with each other partially with a very low probability. Moreover, these binary files comprise the biggest portion of the total amount of traffic in the Internet [61]. Considering these two Internet traffic properties, a traffic-type based bypass technique [55, 57] was proposed to reduce the computational overhead incurred by DRE while keeping the comparable bandwidth saving gains with the bypass technique disabled. This traffic-type based bypass technique incorporates deep packet inspection (DPI). The bypass module tracks the traffic types of each flow by parsing the HTTP header for the "mime-type" header field, and marks any flow as "binary-flow" whose content has been identified as "audio", "video", or "application" in the header field of "mime-type". All of the flows marked as "binary-flow" are passed without any redundancy elimination processing. By applying this traffic-type based DRE bypass technique to the traffic load mixed with text and binary files, the CPU processing load can be reduced greatly while achieving the comparable bandwidth saving gains with this bypass technique disabled. At the same time, the total numbers of memory and storage access are also decreased since fewer data are processed for redundancy elimination.

B. Non-uniform sampling

The non-uniform chunk popularity [60] enables a non-uniform sampling technique to improve DRE performance. The chunks with high-redundant contents are preferred for redundancy elimination than the chunks with low-redundant contents. Thus, DRE performance can be improved by introducing multiple sampling rates and adjusting the sampling rate according to the chunk content rather than with a single sampling rate. In general, text data is more redundant than non-text data [59, 60]. In [4], a non-uniform sampling approach was proposed based on the proportion of plain-text characters within a data chunk: higher sampling rate is applied to the data content that has higher proportion of text characters.

C. Overlap

As investigated in [4], allowing chunk overlap can improve bandwidth savings by 9-14% with relative cost in execution time. Allowing any overlap may result in unacceptable execution time; hence, half-chunk overlap is suggested as a threshold for chunk overlapping.

VI. Conclusion

In this paper, we have reviewed current state of the art of protocol-independent redundancy elimination. We have detailed the system architecture and main processing procedures incurred in protocol-independent DRE techniques. The major mechanisms involved in DRE techniques, including fingerprinting, cache management, chunk matching, and decoding error recovery, have been discussed. For each mechanism, different approaches have been reviewed. We have also presented several redundancy elimination systems deployed in wireline, wireless and cellular networks, respectively. Moreover, some other techniques to enhance the DRE performance such as DRE bypass techniques, non-uniform sampling, and chunk overlap, have been discussed. This survey enables researchers and practitioners to jump-start their DRE projects. Fast and efficient redundant content identification methods to reduce the computational cost involved in identifying the redundant contents and to improve the redundant content elimination efficiency are extremely critical and still hot topics for DRE research. Also, the balance between the computational cost and redundant content elimination efficiency has not been sufficiently addressed in the previous works and needs further studies. In Reference [62], a redundancy elimination method has been proposed to remove duplicated contents at the block level by applying aspects of several techniques, including duplicate block elimination, delta encoding and data compression. Particularly, it first suppresses the identical blocks, and then performs delta encoding on similar blocks with sufficient redundancy, and finally compresses the remaining blocks that do not benefit from the delta encoding. The performance of the proposed redundancy elimination method was, however, not provided in [62]. While individual redundancy elimination techniques such as compression, delta encoding, caching and DRE techniques may be proven to be effective within their respective application scenarios, further investigation is needed to collectively incorporate and exploit multiple techniques in building a complete system.

References

[1] N. T. Spring and D. Wetherall, "A Protocol-Independent Technique for Eliminating Redundant Network Traffic," in *Proc. of SIGCOMM*, Stockholm, Sweden, 2000, pp. 87–95.

[2] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, "Redundancy in Network Traffic: Findings and Implications," in *Proc. of the 11th International Joint Conference on Measurement and Modeling of Computer Systems*, Seattle, WA, USA, 2009, pp. 37–48.

[3] B. Aggarwal, A. Akella, A. Anand, A. Balachandran, P. Chitnis, C. Muthukrishnan, R. Ramjee, and G. Varghese, "EndRE: an End-System Redundancy Elimination Service for Enterprises," in *Proc. of the 7th USENIX conference on Networked systems design and implementation*, San Jose, California, 2010, pp. 28–28.

[4] E. Halepovic, C. Williamson, and M. Ghaderi, "Enhancing Redundant Network Traffic Elimination," *Computer Networks*, vol. 56, no. 2, pp. 795–809, Feb. 2012.

[5] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2010-2015," http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf.

[6] W. J. Bolosky, S. Corbin, D. Goebel, and J. R. Douceur, "Single Instance Storage in Windows 2000," in *Proc. of the 4th conference on USENIX Windows Systems Symposium*, Washington, Aug. 2000, pp. 13–24.

[7] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming Space from Duplicate Files in a Serverless Distributed File System," in *Proc. of the 22 nd International Conference on Distributed Computing Systems*, Vienna, Austria, July 2002, pp. 617–624.

[8] Q. He, Z. Li, and X. Zhang, "Data Deduplication Techniques," in *Proc. of International Conference on Future Information Technology and Management Engineering (FITME)*, Oct. 2010, pp. 430 –433.

[9] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet Caches on Routers: the Implications of Universal Redundant Traffic Elimination," in *Proc. of SIGCOMM*, Seattle, WA, USA, 2008, pp. 219–230.

[10] A. Anand, V. Sekar, and A. Akella, "SmartRE: an Architecture for Coordinated Network-Wide Redundancy Elimination," *SIGCOMM Computer Communication Review*, vol. 39, pp. 87–98, August 2009.

[11] G. Lu, Y. Jin, and D. Du, "Frequency Based Chunking for Data De-Duplication," in *Proc. of Modeling, Analysis Simulation of Computer and Telecommunication Systems*, Aug. 2010, pp. 287 –296.

[12] S. Saha, A. Lukyanenko, and A. Yla-Jaaski, "Combiheader: Minimizing the number of shim headers in redundancy elimination systems," in *Proc. of INFOCOM workshops*, April 2011, pp. 798 –803.

[13] S. Quinlan and S. Dorward, "Venti: A New Approach to Archival Data Storage," in *Proc. of the 1st USENIX Conference on File and Storage Technologies (FAST)*, 2002, pp. 89 – 101.

[14] A. T. Clements, I. Ahmad, M. Vilayannur, and J. Li, "Decentralized Deduplication in SAN Cluster File Systems," in *Proc. of USENIX Annual technical conference*, San Diego, California, June 2009.

[15] C. Constantinescu, J. Pieper, and T. Li, "Block Size Optimization in Deduplication Systems," in *Proc. of Data Compression Conference*, Snowbird, Utah, March 16-18, 2009.

[16] E. Lev-Ran, I. Cidon, and I. Z. Ben-Shaul, "Method and Apparatus for Reducing Network Traffic over Low Bandwidth Links," Patent US 7 636 767, November 2009, File: November 2005.

[17] Juniper Networks, "Application Acceleration," http://www.juniper.net/us/en/products-services/application-acceleration/.

[18] BlueCoat Systems, "WAN Optimization," http://www.bluecoat.com/products.

[19] S. McCanne and M. J. Demmer, "Content-based Segmentation Scheme for Data Compression in Storage and Transmission Including Hierarchical Segment Representation," Patent US 6 828 925, December 2004, File: December 2003.

[20] M. Al-laham and I. M. M. E. Emary, "Comparative study between various algorithms of data compression techniques," *International Journal of Computer Science and Network Security*, vol. 7, no. 4, April 2007.

[21] D. Zeng, F.-Y. Wang, and M. Liu, "Efficient Web Content Delivery Using Proxy Caching Techniques," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 34, no. 3, pp. 270–280, Aug. 2004.

[22] PeerApp, "Video and P2P Caching," http://www.peerapp.com/Technology/VideoCaching.aspx.

[23] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy, "Organization-based Analysis of Web-object Sharing and Caching," in *Proc. of the 2nd conference on USENIX Symposium on Internet Technologies and Systems (USITS'99)*, Boulder, Colorado, October 1999, pp. 25–36.

[24] "Squid Web proxy cache," http://www.squid-cache.org/.

[25] B. C. Housel and D. B. Lindquist, "WebExpress: a System for Optimizing Web Browsing in a Wireless Environment," in *Proc. of the 2nd annual International Conference on Mobile Computing and Networking (MobiCom'96)*, November 1996.

[26] J. C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy, "Potential Benefits of Delta Encoding and Data Compression for HTTP," in *Proc. of the ACM SIGCOMM*, Cannes, France, September 1997, pp. 181–194.

[27] F. Douglis and A. Iyengar, "Application-specific Delta-encoding via Resemblance Detection," in *Proc. of USENIX Annual Technical Conference*, San Antonio, TX, June 2003, pp. 113–126.

[28] Rsync, http://rsync.samba.org/.

[29] N. Mandagere, P. Zhou, M. A. Smith, and S. Uttamchandani, "Demystifying Data Deduplication," in *Proc. of the Middleware Conference Companion*, Leuven, Belgium, Dec. 1-5, 2008, pp. 12–17.

[30] Y. Song, K. Guo, and L. Gao, "Redundancy-Aware Routing with Limited Resources," in *Proc. of the 19th International Conference on Computer Communications and Networks (ICCCN)*, Aug. 2010, pp. 1–6.

[31] Z. Zhuang and R. Sivakumar, "Wireless Memory: Eliminating Communication Redundancy in Wi-Fi Networks," in *Proc. of IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Lucca, Tuscany, Italy, June 20-24, 2011.

[32] E. Zohar, I. Cidon, and O. O. Mokryn, "Celleration: Loss-Resilient Traffic Redundancy Elimination for Cellular Data," in *Proc. of the 13th Internaltional Workshop on Mobile Computing Systems and Applications (HotMobile)*, San Diego, California, Feb. 28-29, 2012, pp. 10:1–10:6.

[33] S.-H. Shen, A. Gember, A. Anand, and A. Akella, "REfactor-ing Content Overhearing to Improve Wireless Performance," in *Proc. of the 17th Annual International Conference on Mobile Computing and Networking*, Las Vegas, Nevada, USA, 2011, pp. 217–228.

[34] C. Lumezanu, K. Guo, N. Spring, and B. Bhattacharjee, "The Effect of Packet Loss on Redundancy Elimination in Cellular Wireless Networks," in *Proc. of the 10th annual Conference on Internet Measurement (IMC)*, Melbourne, Australia, 2010, pp. 294–300.

[35] E. Zohar, I. Cidon, and O. O. Mokryn, "The Power of Prediction: Cloud Bandwidth and Cost Reduction," in *Proc. of the SIGCOMM*, Toronto, Canada, Aug. 15-19, 2011.

[36] E. Halepovic, M. Ghaderi, and C. Williamson, "On the Performance of Redundant Traffic Elimination in WLANs," in *Proc. of IEEE International Conference on Communications*, Ottawa, Canada, June 2012.

[37] M. Martynov, "Challenges for High-Speed Protocol-Independent Redundancy Eliminating Systems," in *Proc. of 18th Internatonal Conference on Computer Communications and Networks (ICCCN)*, Aug. 2009, pp. 1 –6.

[38] U. Manber, "Finding Similar Files in a Large File System," in *Proc. of the USENIX Winter 1994 Technical Conference (WTEC'94)*, San Francisco, California, January 1994.

[39] A. Broder, "On the Resemblance and Containment of Documents," in *Proc. of the Compression and Complexity of Sequences (SEQUENCES'97)*, Washington, DC, USA, 1997, pp. 21–29.

[40] M. O. Rabin, "Fingerprinting by Random Polynomials," Center for Research in Computing Technology, Harvard University, Tech. Rep. TR-15-81, 1981.

[41] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local Algorithms for Document Fingerprinting," in *Proc. of the 2003 ACM SIGMOD international conference on Management of Data (SIGMOD)*, San Diego, California, 2003, pp. 76–85.

[42] N. Bjørner, A. Blass, and Y. Gurevich, "Content-Dependent Chunking for Differential Compression, the Local Maximum Approach," http://research.microsoft.com/en-us/um/people/gurevich/Opera/190.pdf, Microsoft Research, Tech. Rep. 109, July 2007.

[43] "Jenkins Hash," http://burtleburtle.net/bob/c/lookup3.c.

[44] E. Halepovic, C. Williamson, and M. Ghaderi, "DYNABYTE: A Dynamic Sampling Algorithm for Redundant Content Detection," in *Proc. of the 20th International Conference on Computer Communications and Networks (ICCCN)*, Maui, Hawaii, USA, July 31 - Aug. 4, 2011.

[45] M. G. Emir Halepovic, Carey Williamson, "Low-Overhead Dynamic Sampling for Redundant Traffic Elimination," vol. 7, no. 1, Jan. 2012.

[46] S. Ihm, K. Park, and V. S. Pai, "Wide-area Network Acceleration for the Developing World," in *Proc. of the 2010 USENIX conference on USENIX annual technical conference*, Boston, MA, 2010, pp. 18–18.

[47] E. Zohar, O. O. Mokryn, and I. Cidon, "PACK: Speculative TCP Traffic Redundancy Elimination," Tech. Rep. CCIT Report #770, July 2010. [Online]. Available: http://webee.technion.ac.il/publication-link/index/id/569/abstract/1.

[48] D. G. Thaler and C. V. Ravishankar, "Using Name-based Mappings to Increase Hit Rates," *IEEE/ACM Transaction on Networking*, vol. 6, no. 1, pp. 1–14, Feb. 1998.

[49] J. Luo, D. Ye, X. Liu, and M. Fan, "A survey of Multicast Routing Protocols for Mobile Ad-Hoc Networks," *IEEE Communications Surveys Tutorials*, vol. 11, no. 1, pp. 78 –91, Quarter 2009.

[50] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen, "cSamp: A System for Network-Wide Flow Monitoring," in *Proc. 5th USENIX NSDI*, San Francisco, CA, Apr. 2008.

[51] I. Papapanagiotou, R. D. Callaway, and M. Devetsikiotis, "Chunk and Object Level Deduplication for Web Optimization: A Hybrid Approach," in *Proc. of IEEE International Conference on Communications (ICC)*, Ottawa, Canada, June 10-15, 2012.

[52] M. Martynov, "Experimental Study of Protocol-independent Redundancy Elimination Algorithms," in *Proc. of the first joint WOSP/SIPEW international conference on Performance Engineering*, San Jose, California, USA, 2010, pp. 141–146.

[53] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "RFC 2616: Hypertext Transfer Protocol – HTTP/1.1," June 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

[54] K. Tangwongsan, H. Pucha, D. G. Andersen, and M. Kaminsky, "Efficient Similarity Estimation for Systems Exploiting Data Redundancy," in *Proc. of the 29th conference on Information Communications (INFOCOM'10)*, San Diego, California, USA, 2010, pp. 1487–1495.

[55] S. Saha, "On Reducing the Processing Load of Redundancy Elimination Algorithms," in *Proc. of IEEE GLOBECOM Workshops*, Dec. 2011, pp. 1106 –1110.

[56] W. Xia, H. Jiang, D. Feng, and L. Tian, "Accelerating Data Deduplication by Exploiting Pipelining and Parallelism with Multicore or Manycore Processors," in *Proc. of the 10th USENIX Conference on File and Storage Technologies (FAST)*, San Jose, CA, USA, Feb. 14-17, 2012.

[57] Y. Zhang, N. Ansari, M. Wu, and H. Yu, "SDRE: Selective data redundancy elimination for resource constrained hosts," in *Proc. of 2012 IEEE Global Communications Conference (GLOBECOM)*, Anaheim, CA, USA, 3-7 Dec. 2012, pp. 1865–1870.

[58] M. Arlitt and C. Williamson, "Internet Web servers: Workload Characterization and Performance Implications," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 631–645, Oct. 1997.

[59] H. Pucha, D. G. Andersen, and M. Kaminsky, "Exploiting Similarity for Multi-source Downloads Using File Handprints," in *Proc. of the 4th USENIX conference on Networked Systems Design & Implementation (NSDI'07)*, Cambridge, MA, 2007.

[60] E. Halepovic, C. Williamson, and M. Ghaderi, "Exploiting Non-Uniformities in Redundant Traffic Elimination," Department of Computer Science, University of Calgary, Tech. Rep., 2010. [Online]. Available: https://dspace.ucalgary.ca/bitstream/1880/48155/1/2010-971-20.pdf

[61] M. E. Crovella, M. S. Taqqu, and A. Bestavros, "Heavy-tailed Probability Distributions in the World Wide Web," in *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, 1998, pp. 3–25.

[62] F. Douglis, P. Kulkarni, J. D. LaVoie, and J. M. Tracey, "Method and Apparatus for Data Redundancy Elimination at the Block Level," Patent US 20 050 131 939 A1, Jun. 16, 2005, Filed on Dec. 16, 2003.

Yan Zhang (S'10) received the B.E. and M.E. degrees in Electrical Engineering from Shandong University, Jinan, Shandong, China, in 2001 and 2004, respectively. She is currently pursuing the Ph.D. degree in Computer Engineering at the New Jersey Institute of Technology (NJIT), Newark. Her research interests include automatic modulation classification algorithms, distributed detection in sensor network, congestion control in data center networks, content delivery acceleration over wide area networks, and energy-efficient networking.

Nirwan Ansari (S'78-M'83-SM'94-F'09) received his BSEE (summa cum laude with a perfect GPA)from NJIT, MSEE from the University of Michigan, Ann Arbor, and PhD from Purdue University, West Lafayette, IN, USA. He joined NJIT in 1988, where he is a Professor of electrical and computer engineering. He has also assumed various administrative positions at NJIT. He has been a Visiting (Chair/Honorary) Professor at several universities. His current research focuses on various aspects of broadband networks and multimedia communications.

Prof. Ansari has served on the editorial/advisory boards of eight journals. He was elected to serve on the IEEE Communications Society (ComSoc) Board of Governors as a member-at-large (2013-2015) as well as the IEEE Region 1 Board of Governors as the IEEE North Jersey Section Chair. He has chaired ComSoc technical committees, and has been actively organizing numerous IEEE international conferences/symposia/workshops, assuming leadership roles as Chair or TPC Chair.

Prof. Ansari has authored Computational Intelligence for Optimization (Springer 1997) with E.S.H. Hou, Media Access Control and Resource Allocation for Next Generation Passive Optical Networks (Springer, 2013) with J. Zhang, and edited Neural Networks in Telecommunications (Springer1994) with B. Yuhas. He has also contributed to over 400 publications, over one third of which were published in widely cited refereed journals/magazines. He has been granted over twenty U.S. patents. He has also guest-edited a number of special issues, covering various emerging topics in communications and networking. He has been frequently selected to deliver keynote addresses, distinguished lectures, and tutorials. Some of his recent recognitions include Best Paper awards, Excellence in Teaching Awards, the Thomas Alva Edison Patent Award (2010), the NJ Inventors Hall of Fame Inventor of the Year Award (2012), and designation as a ComSoc Distinguished Lecturer (2006-2009).