ISSN: 2321-7782 (Online)
Volume 3, Issue 9, September 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Prediction Analysis for Cloud Bandwidth and Cost Reduction

Ankita Kotalwar[1]
M.E(CSE) Student, MPGI College of Engineering
Nanded, India

Dr. Sadhana Chidrawar[2]
Dean's MPGI College of Engineering
Nanded, India

Abstract: ***In this paper, we present PACK (Predictive ACKs), a novel end-to-end traffic redundancy elimination (TRE) system, designed for cloud computing customers. Cloud based TRE needs to apply a judicious use of cloud resources so that the bandwidth cost reduction combined with the additional cost of TRE computation and storage would be optimized. PACK's main advantage is its capability of offloading the cloud- server TRE effort to end-clients, thus minimizing the processing costs induced by the TRE algorithm. Unlike previous solutions, PACK does not require the server to continuously maintain clients' status. This makes PACK very suitable for pervasive computation environments that combine client mobility and server migration to maintain cloud elasticity. PACK is based on a novel TRE technique, which allows the client to use newly received chunks to identify previously received chunk chains, which in turn can be used as reliable predictors to future transmitted chunks. We present a fully functional PACK implementation, transparent to all TCP-based applications and net-work devices. Finally, we analyze PACK benefits for cloud users, using traffic traces from various sources.***

Keywords: Caching, Cloud Computing, Network Optimization, Traffic Redundancy Elimination

I. INTRODUCTION

The data centre hardware and software is what we will call a cloud. When a cloud is made available in a pay-as you- go manner to the general public, we call it a public cloud; the service being sold is utility computing. We use the term private cloud to refer to internal data centres of a business or other organization, not made available to the general public, when they are large enough to benefit from the advantages of cloud computing that we discuss here. Thus, cloud computing is the sum of SaaS and utility computing, but does not include small or medium sized data centres, even if these rely on virtualization for management. People can be users or providers of SaaS, or users or providers of utility computing. We focus on SaaS providers (cloud users) and cloud providers, which have received less attention than SaaS users. In some cases, the same actor can play multiple roles. For instance, a cloud provider might also host its own customer-facing services on cloud infrastructure. From a hardware provisioning and pricing point of view, three aspects are new in cloud computing. The appearance of infinite computing resources available on demand, quickly enough to follow load surges, thereby eliminating the need for cloud computing users to plan far ahead for provisioning. The elimination of an up-front commitment by cloud users, thereby allowing companies to start small and increase hardware resources only when there is an increase in their needs. The ability to pay for use of computing resources on a short-term basis as needed (for example, processors by the hour and storage by the day) and release them as needed, thereby rewarding conservation by letting machines and storage go when they are no longer useful. Cloud computing is emerging style of delivery in which applications, data and resources are rapidly provisioned as standardized offerings to users with a flexible price. The cloud computing paradigm has achieved widespread adoption in recent years. Its success is due largely to customers' ability to use services on demand with a pay-as-you go [6] pricing model, which has proved convenient in many respects. Low costs and high flexibility make migrating to the cloud compelling. Cloud computing is the long dreamed vision of computing as a utility, where users can remotely store their data into the cloud so as to enjoy the on demand high quality applications and services from a shared pool of configurable computing resources. By data outsourcing, users can be relieved from the burden of local data storage and maintenance. Traffic redundancy and elimination approach is used for

minimizing the cost. Traffic redundancy stems from common end-users' activities, such as repeatedly accessing, downloading, distributing and modifying the same or similar information items (documents, data, web and video). TRE is used to eliminate the transmission of redundant content and therefore, to significantly reduce the network cost. In most common TRE solutions, both the sender and the receiver ex- amine and compare signatures of data chunks, parsed according to the data content prior to their transmission. When redundant chunks are detected, the sender replaces the transmission of each redundant chunk with its strong signature [20]. Commercial TRE solutions are popular at enterprise networks and the deployment of two or more proprietary protocol, state synchronized middleboxes at both the intranet entry points of data centres and branch offices, eliminating repetitive traffic between them.While proprietary middle-boxes are popular point solutions within enterprises, they are not as attractive in a cloud environment. First, cloud providers cannot benefit from a technology whose goal is to reduce customer bandwidth bills, and thus are not likely to invest in one. Moreover, a fixed client-side and server-side middle-box pair solution is inefficient for a combination of a mobile environment, which detaches the client from a fixed location, and cloud-side elasticity which motivates work distribution and migration among data centres. Therefore, it is commonly agreed that a universal, software-based, end-to-end TRE is crucial in today's pervasive environment [4,1]. This enables the use of a standard protocol stack and makes a TRE within end-to-end secured traffic (e.g., SSL) possible. In this paper, we show that cloud elasticity calls for a new TRE solution that does not require the server to continuously maintain clients' status. First, cloud load balancing and power optimizations may lead to a server-side process and data migration environment, in which TRE solutions that require full synchronization between the server and the client are hard to accomplish or may lose efficiency due to lost synchronization.

II. Related Work

Several TRE techniques have been explored in recent years. A protocol-independent TRE was proposed in [4]. The paper describes a packet-level TRE, utilizing the algorithms presented in [16]. Several commercial TRE solutions described in [15] and [18], have combined the sender-based TRE ideas of [4] with the algorithmic and implementation approach of [20] along with protocol specific optimizations for middleboxes solutions. In particular, [15] describes how to get away with three-way handshake between the sender and the receiver if a full state synchronization is maintained. [3] and [5] present redundancy-aware routing algorithm. These papers assume that the routers are equipped with data caches, and that they search those routes that make a better use of the cached data. A large-scale study of real-life traffic redundancy is presented in [11] and [4]. Our paper builds on the latter's finding that "an end to end redundancy elimination solution, could obtain most of the middle-box's bandwidth savings", motivating the benefit of low cost software end-to-end solutions. Wanax [12] is a TRE system for the developing world where storage and WAN bandwidth are scarce. It is a software-based middle-box replacement for the expensive commercial hardware. In this scheme, the sender middle-box holds back the TCP stream and sends data signatures to the receiver middle-box. The receiver checks whether the data is found in its local cache. Data chunks that are not found in the cache are fetched from the sender middle-box or a nearby receiver middle-box. Naturally, such a scheme incurs a three-way-handshake latency for non-cached data. EndRE [1] is a sender-based end-to-end TRE for enterprise net- works. It uses a new chunking scheme that is faster than the commonly- used Rabin fingerprint, but is restricted to chunks as small as 32-64 bytes. Unlike PACK, EndRE requires the server to maintain a fully and reliably synchronized cache for each client. To adhere with the server's memory requirements these caches are kept small making the system inadequate for medium-to-large content or long-term redundancy. EndRE is server specific, hence not suitable for a CDN or cloud environment. To the best of our knowledge none of the previous works have addressed the requirements for a cloud computing friendly, end-to- end TRE which forms PACK's focus.

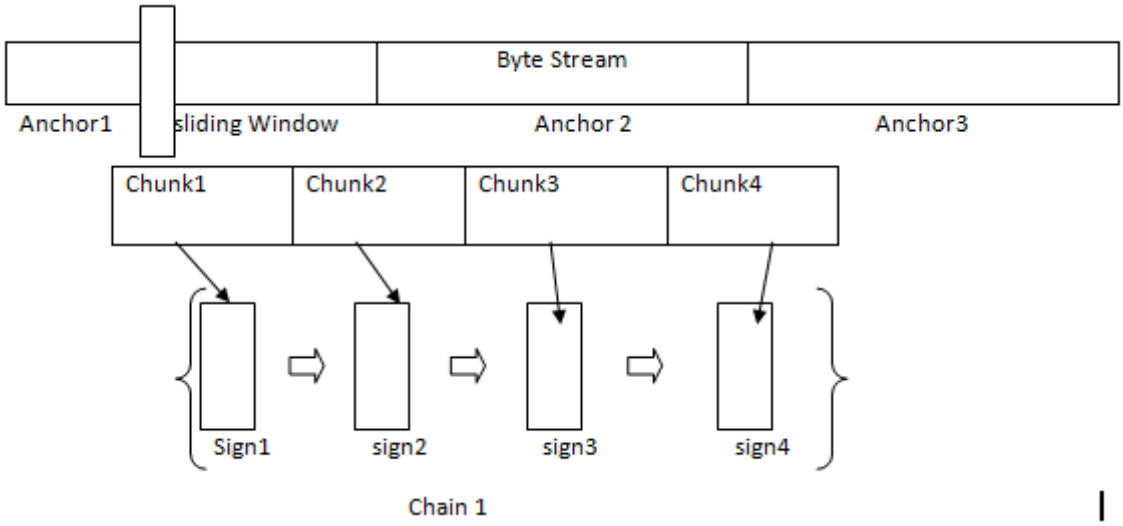


Fig1. From stream of Chain

Chunk Size

The utilization of a small chunk size presents better redundancy elimination when data modifications are fine-grained, such as sporadic changes in a HTML page. On the other hand, the use of smaller chunks increases the storage index size, memory usage and magnetic disk seeks. It also increases the transmission overhead of the virtual data exchanged between the client and the server.

III. THE PACK ALGORITHM

We first describe the basic receiver- driven operation of the PACK protocol. Several enhancements and optimizations are introduced. The stream of data received at the PACK receiver is parsed to a sequence of variable size, content-based signed chunks similar to [16][20]. The chunks are then compared to the receiver local storage,termed chunk store. If a matching chunk is found in the local chunk store, the receiver retrieves the sequence of subsequent chunks, referred to as a chain, by traversing the sequence of LRU chunk pointers that are included in the chunks' metadata. Using the constructed chain, the receiver sends a prediction to the sender for the subsequent data. Part of each chunk's prediction, termed a hint, is an easy to compute function with a small enough false-positive value, such as the value of the last byte in the predicted data or a byte-wide XOR checksum of all or selected bytes. The prediction sent to the receiver includes the range of the predicted data, the hint and the signature of the chunk. The sender identifies the predicted range in its buffered data, and verifies the hint for that range. If the result matches the received hint, it continues to perform the more computationally intensive SHA-1 signature operation. Upon a signature match, the sender sends a confirmation message to the receiver, enabling it to copy the matched data from its local storage.

A) Receiver Side Chunk Storage

Predictive ACK uses the new continuous chains scheme that described in Fig. 1, in that every chunk are related to all other chunks by their recent received way of order. The Predictive ACK receivers have to keep a chunk storage, which it's a very large size cache of chunks and their metadata. Chunk's metadata includes the data chunk's signature and a single pointer to the successive chunk in the recent received stream that contain this chunk. Cache and index technique are employed to efficiently maintain and retrieve the every stored chunks and its signature and the chains created by traverse the chunk pointers.

B) Receiver Side Algorithm

The arrival of a new information, the receiver side system that respective signature for every chunk and see the match in its local (temporary) chunk storage. If the chunk's match is founded, the receiver side determines whether it's a part of a formerly received chunk chain, using the chunks' metadata (data about data) otherwise in affirmative, the receiver send a prediction to

the sender side for the several new expected chain chunks. It carries a beginning point in the byte stream that is offset and the identity of several subsequent chunks.

Proc. 1: Receiver Segment Processing

Proc. 2: predAttempt()

Proc. 3: processPredAck()

C) Sender Side Algorithm

Sender receives a Predictive message from the receiver side and it tries to compare the received predictions to its buffered yet to be sent information. For every prediction, the sender has to determine that corresponding TCP range of sequence and verifies it. If that hint match, the sender measures the more computationally intensive Secure Hash Algorithm- 1 signature for the predicted information range and match the result to the signature received in the Predictive message of data. In this case if the hint does not same, a computationally expansive operation is saved. If the two Secure Hash Algorith-1 signatures compare, the sender can safely assume that the receiver's prediction method is absolutely correct and, it replace the entire outgoing buffered data with a Predictive ACK message.

IV. OPTIMIZATIONS

A) Adaptive Receiver Virtual Window

Predictive ACK enable the receiver side to locally capture the sender data when a local or temporary copy is available, thus eliminating the requirement to send this information through the network. In this term the receiver's fetching of that recent local data as the reception of visual data.

B) Cloud Server Acting as a Receiver

In a developing trend, cloud computing storage is getting a dominant player from backup of store and sharing of data services to the American National Library and e –mail services. In this most of these Services, the cloud is used often the receiver of the data.

C) Hierarchal Approach

Predictive ACK's receiver side based mode is less amount of efficient if changes in the information are scattered. In this scenario, the prediction continuation are frequently interrupted, In this turn, forces the sender to retransmit to the raw data transmission until a new comparison is found at the receiver side and It reported back to the sender Side. To that end.

V. MOTIVATING A RECEIVER-BASED APPROACH

The objective of this section is twofold: evaluating the potential data redundancy for several applications that are likely to reside in a cloud, and to estimate the PACK performance and cloud costs of the redundancy elimination process. Our evaluations are conducted using: 1) video traces captured at a major ISP; 2) traffic obtained from a popular social network service; and 3) genuine data sets of real-life workloads. In this section, we relate to an average chunk size of 8 kB, although our algorithm allows each client to use a different chunk size.

Traffic Redundancy

1) Traffic Traces: We obtained a 24-h recording of traffic at an ISP's 10-Gb/s PoP router, using a 2.4-GHz CPU recording machine with 2 TB storage (4 500 GB 7 200 RPM disks) and 1-Gb/s NIC. We filtered YouTube traffic using deep packet inspection and mirrored traffic associated with YouTube servers IP addresses to our recording device. Our measurements show that YouTube traffic accounts for 13% of the total daily Web traffic volume of this ISP. The recording of the full YouTube stream would require 3 times our network and disk write speeds. Therefore, we isolated 1/6 of the obtained YouTube traffic,

grouped by the video identifier (keeping the redundancy level intact) using a programmed load balancer that examined the upstream HTTP requests and redirected downstream sessions according to the video identifier that was found in the YouTube's URLs, to a total of 1.55 TB. For accurate reading of the true redundancy, we filtered out the client IP addresses that were used too intensively to represent a single user and were assumed to represent a NAT address. Note that YouTube's video content is not cacheable by standard Web proxies since its URL contains private single-use tokens changed with each HTTP request. Moreover, most Web browsers cannot cache and reuse partial movie downloads that occur when end-users skip within a movie or switch to another movie before the previous one ends. Table I summarizes our findings. We recorded more than 146 K distinct sessions, in which 37 K users request over 39 K distinct movies. Average movie size is 15 MB, while the

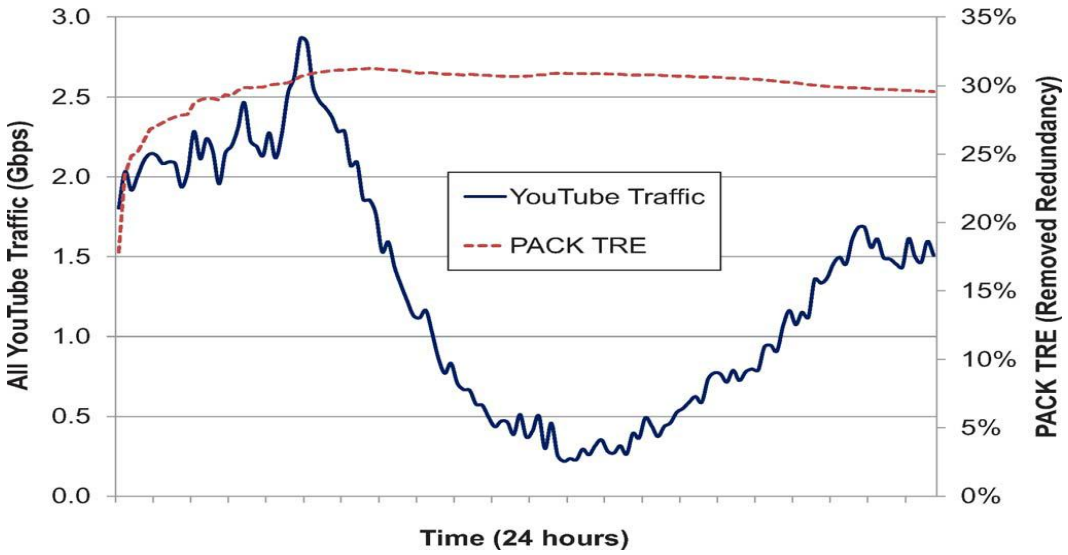


Fig. 2. ISP's YouTube traffic over 24 h, and PACK redundancy elimination ratio with this data.

Average session size is 12 MB, with the difference stemming from end-user skips and interrupts. When the data is sliced into 8-kB chunks, PACK brings a traffic savings of up to 30%, assuming the end-users start with an empty cache, which is a worst-case scenario. Fig.2 presents the YouTube traffic and the redundancy obtained by PACK over the entire period, with the redundancy sampled every 10 min and averaged. This end-to-end redundancy arises solely from self-similarity in the traffic created by end-users. We further analyzed these cases and found that end-users very often download the same movie or parts of it repeatedly. The latter is mainly an intersession redundancy produced by end-users that skip forward and backward in a movie and producing several (partially) overlapping downloads. Such skips occurred at 15% of the sessions and mostly in long movies (over 50 MB). Since we assume the cache is empty at the beginning, it takes a while for the chunk cache to fill up and enter a steady state. In the steady state, around 30% of the traffic is identified as redundant and removed. We explain the length of the warm-up time by the fact that YouTube allows browsers to cache movies for 4 h, which results in some replays that do not produce downloads at all.

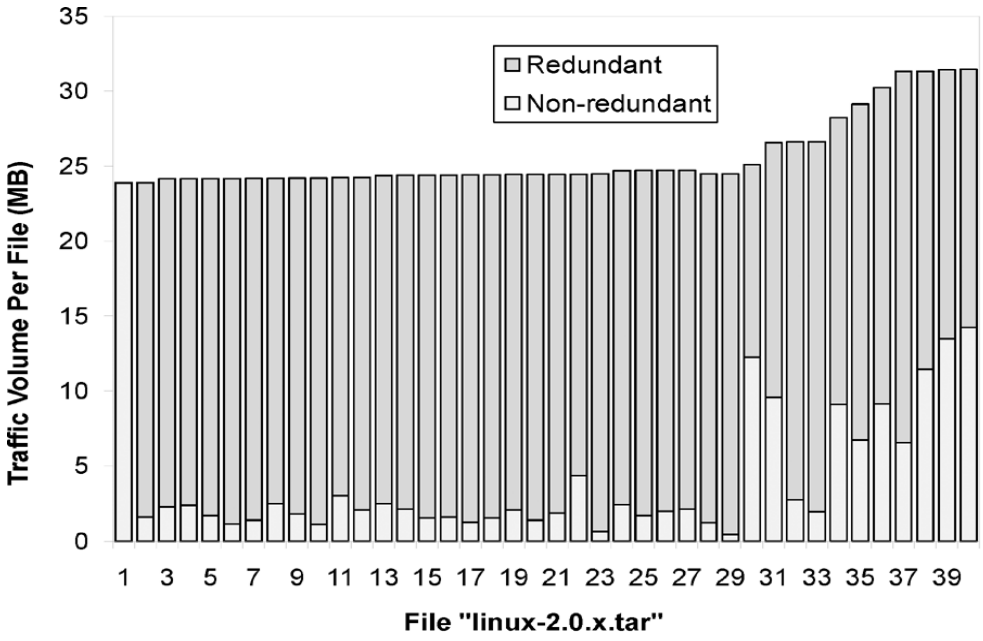
2) Static Dataset: We acquired the following static datasets:

Linux source—different Linux kernel versions: all the 40 2.0.x tar files of the kernel source code that sum up to 1 GB;

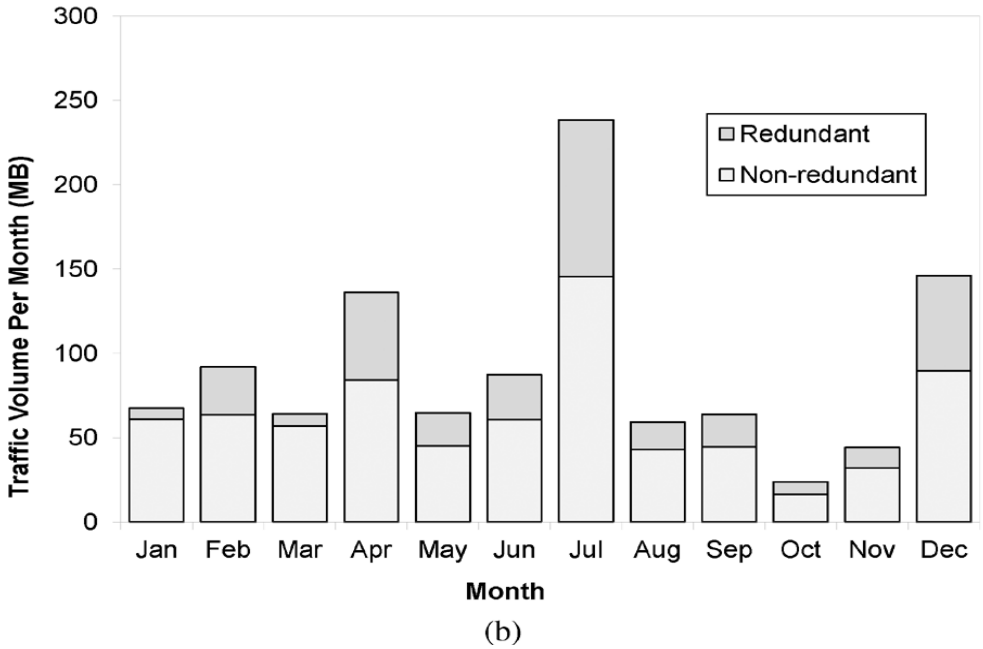
Email—a single-user Gmail account with 1140 e-mail messages over a year that sum up to 1.09 GB. The 40 Linux source versions were released over a period of 2 years. All tar files in the original release order, from 2.0.1 to 2.0.40, were downloaded to a download directory, mapped by PACK, to measure the amount of redundancy in the resulted traffic. Fig. 3(a) shows the redundancy in each of the downloaded versions. Altogether, the Linux source files show 83.1% redundancy, which accounts to 830 MB. To obtain an estimate of the redundancy in e-mail traffic, we operated an IMAP client that fully synchronized the remote Gmail account with a new local folder. Fig. 3(b) shows the redundancy in each month, according to the e-mail message's issue date. The total measured traffic redundancy was 31.6%, which is roughly 350 MB. We found this redundancy to arise from large attachments that are sent by multiple sources, e-mail correspondence with similar documents in development

process, and replies with large quoted text. This result is a conservative estimate of the amount of redundancy in cloud e-mail traffic because in practice some messages are read and downloaded multiple times. For example, a Gmail user that reads the same attachment for 10 times, directly from the Web browser, generates 90% redundant traffic. Our experiments show that in order to derive an efficient PACK redundancy elimination, the chunk-level redundancy needs to be applied along long chains. To quantify this phenomenon,

We explored the distribution of redundant chains in the Linux and Email datasets. Fig. 3(a) presents the resulted redundant data chain length distribution. In Linux, 54% of the chunks are found in chains, and in Email about 88%.Moreover, redundant chunks are more probable to reside in long chains. These findings sustain our conclusion that once redundancy is discovered in a single chunk, it is likely to continue in subsequent chunks. Furthermore, our evaluations show that in videos and large files with a small amount of changes, redundant chunks are likely to reside in very long chains that are efficiently handledby a receiver-based TRE.



(a)



(b)

Fig. 3. Traffic volume and detected redundancy. (a) Linux source: 40 different Linux kernel versions. (b) Email: 1-year Gmail account by month.

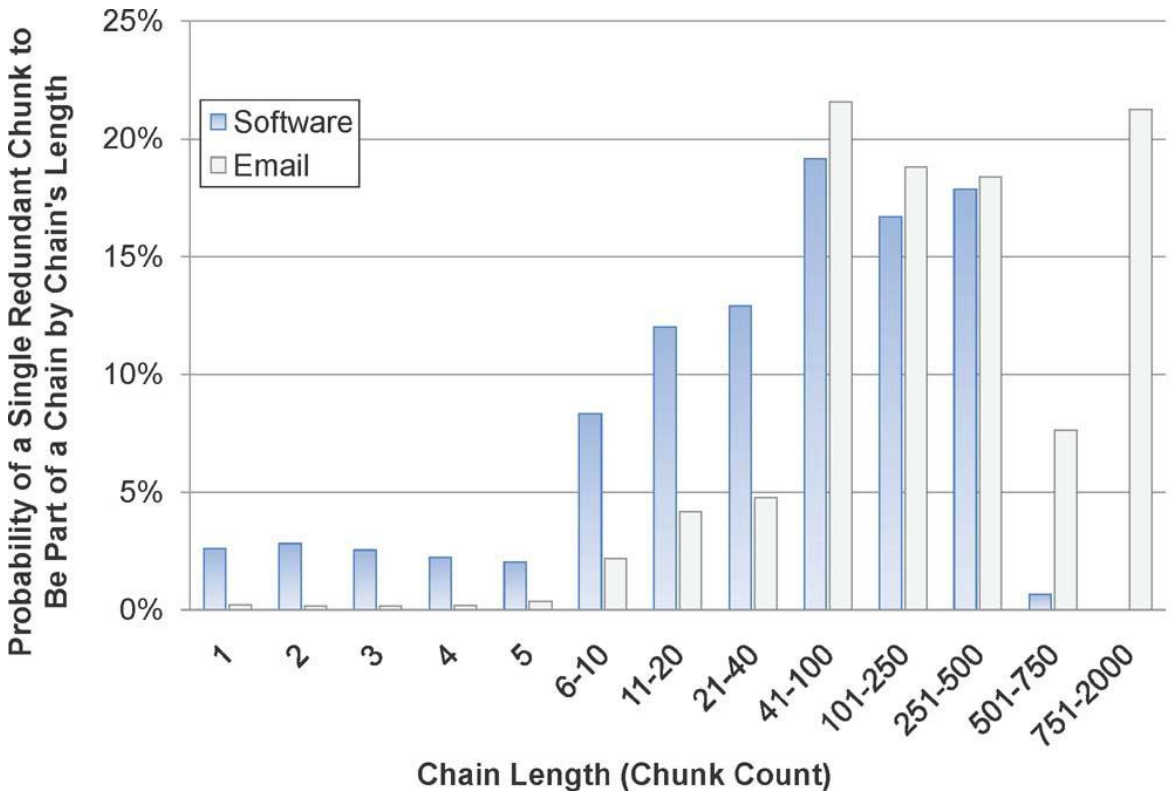


Fig. 4 Chain length histogram Linux Software and Email data collections.

	No TRE	PACK	Server-based
Traffic volume	9.1 TB	6.4 TB	6.2 TB
Traffic cost reduction (Figure 4)		30%	32%
Server-hours cost increase (Figure 9)		6.1%	19.0%
Total operational cost	100%	80.6%	83.0%

Table I Cloud Operational Cost Comparison

Table I summarizes the costs and the benefits of the TRE operations and compares them to a baseline with no TRE. The total operational cost is based on current Amazon EC2 [23] pricing for the given traffic-intensive scenario (traffic: server hours cost ratio of 7:3). Both TRE schemes identify and eliminate the traffic redundancy. However, while PACK employs the server only when redundancy exists, the sender-based TRE employs it for the entire period of time, consuming more servers than PACK and no-TRE schemes when no or little redundancy is detected.

VI. IMPLEMENTATION

In this section, we present PACK implementation, its performance analysis, and the projected server costs derived from the implementation experiments. Our implementation contains over 25 000 lines of C and Java code. It runs on Linux with Netfilter Queue [24]. At the server side, we use an Intel Core 2 Duo 3 GHz, 2 GB of RAM, and a WD1600AAJS SATA drive desktop. The clients laptop machines are based on an Intel Core 2 Duo 2.8 GHz, 3.5 GB of RAM, and a WD2500BJKT SATA drive. Our implementation enables the transparent use of the TRE at both the server and the client. PACK receiver–sender protocol is embedded in the TCP Options field for low overhead and compatibility with legacy systems along the path. We keep the genuine operating systems' TCP stacks intact, allowing a seamless integration with all applications and protocols above TCP.

Chunking and indexing are performed only at the client's side, enabling the clients to decide independently on their preferred chunk size.

PACK chunking algorithm

1. Mask ⇦ 0x00008A3110583080 {48 bytes window; 8 KB chunks}
2. {has to be 64 bits}
3. for all byte stream do
4. shift left *longval* by 1 bit { ; drop msb}
5. bitwise-xor *byte*
6. if processed at least 48 bytes and (*longval* bitwise-and *mask*) then
7. found an anchor
8. end if
9. end for

Algorith for processPredAckHybrid()

1. for all offset PRED-ACK do
2. read data from disk
3. put data in TCP input buffer
4. {new code for Hybrid}
5. for all chunk offset do
6. calcDispersion(0)
7. end for
8. end for

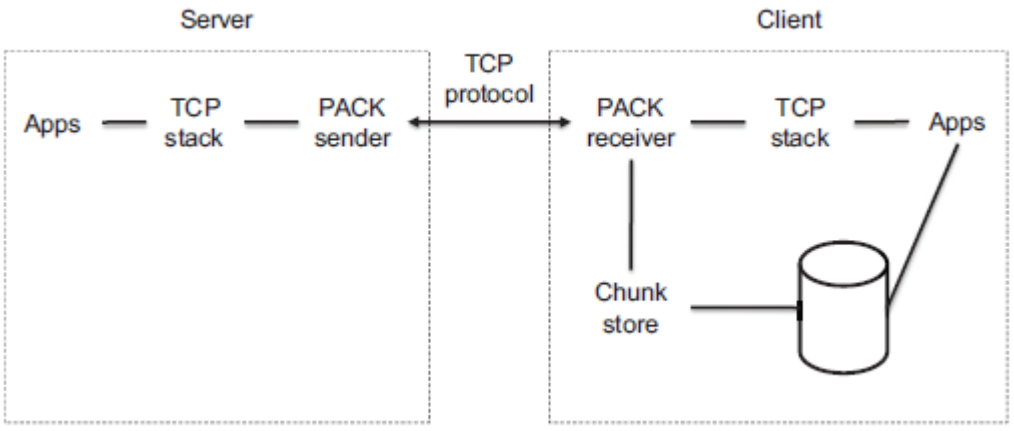


Fig5. Overview of PACK Implementation

We have to present the Predictive ACK hierarchal mode of operation. PACK computes the data dispersion value using an exponential smoothing function

D←Dα+(1-α)M

Where α is a smoothing factor. The value is set to 0 when a chain break is detected, and 255 otherwise.

In our implementation, the client uses an average chunk size of 8 kB. We found this size to achieve high TRE hit-ratio in the evaluated datasets, while adding only negligible overheads of 0.1% in metadata storage and 0.15% in predictions bandwidth. For the experiments held in this section, we generated datasets: IMAP e-mails, HTTP videos, and files downloaded over FTP. The workload was then loaded to the server and consumed by the clients. We sampled the machines' status every second to measure real and virtual traffic volumes and CPU utilization.

VII. Conclusion

The cloud environment redefines the TRE system requirements, making proprietary middle-box solutions inadequate. Consequently, there is a rising need for a TRE solution that reduces the cloud's operational cost while accounting for application latencies, user mobility, and cloud elasticity. In this paper, we have presented PACK, a receiver-based, cloud-friendly, end-to-end TRE that is based on novel speculative principles that reduce latency and cloud operational cost. PACK does not require the server to continuously maintain clients' status, thus enabling cloud elasticity and user mobility while preserving long-term redundancy.

Moreover, PACK is capable of eliminating redundancy based on content arriving to the client from multiple servers without applying a three-way handshake. Our evaluation using a wide collection of content types shows that PACK meets The expected design goals and has clear advantages over sender-based TRE, especially when the cloud computation cost and buffering requirements are important. Moreover, PACK imposes additional effort on the sender only when redundancy is exploited, thus reducing the cloud overall cost. Two interesting future extensions can provide additional benefits to the PACK concept. First, our implementation maintains chains by keeping for any chunk only the last observed subsequent chunk in an LRU fashion. An interesting extension to this work is the statistical study of chains of chunks that would enable multiple possibilities in both the chunk order and the corresponding predictions. The system may also allow making more than one prediction at a time, and it is enough that one of them will be correct for successful traffic elimination. A second promising direction is the mode of operation optimization of the hybrid sender–receiver approach based on shared decisions derived from receiver's power or server's cost changes.

VIII. Future Work

In current paper we cannot remove the small redundant part of the large file content such as formula is not detected so in future we will work on this thing. Even though the most of the content of those large files are not redundant

References

1. E. Zohar, I. Cidon, and O. Mokryn, "The power of prediction: Cloud bandwidth and cost reduction," in Proc. SIGCOMM, 2011, pp. 86–97.
2. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph,R.Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, 2010.
3. U. Manber, "Finding similar files in a large file system,"in Proc. USENIX Winter Tech. Conf., 1994, pp. 1–10.
4. N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in Proc. SIGCOMM, 2000, vol.30, pp. 87–95.
5. A. Muthitacharoen, B. Chen, and D. Mazières, "A low- bandwidth network file system," in Proc. SOSP, 2001, pp.174-187.
6. E. Lev-Ran, I. Cidon, and I. Z. Ben-Shaul, "Method and apparatus for reducing network traffic over low bandwidth links," US Patent 7636767, Nov. 2009.
7. S.Mccanne andM. Demmer, "Content-based segmentation scheme for data compression in storage and transmission including hierarchical segment representation," US Patent 6828925, Dec. 2004.
8. R. Williams, "Method for partitioning a block of data into sub blocks and for storing and communicating such subblocks," US Patent 5990810, Nov. 1999.
9. Juniper Networks, Sunnyvale, CA, USA, "Application acceleration," 1996 [Online]. Available: http://www.juniper.net/us/en/products-services/applicationacceleration/
10. Blue Coat Systems, Sunnyvale, CA, USA, "MACH5," 1996[Online].Available:

11. Expand Networks, Riverbed Technology, San Francisco, CA, USA, "Application acceleration and WAN optimization," 1998 [Online]. Available: http://www.expand.com/technology/application- acceleration.aspx

12. F5, Seattle,WA, USA, "WAN optimization," 1996 [Online]. vailable: http://www. f5.com / solutions / acceleration /wan - optimization/

13. A. Flint, "The next workplace revolution," Nov. 2012 [Online]. Available: http://m.theatlanticcities.com/jobs-and-economy/2012/11/nextworkplace-revolution /3904/

14. A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, "Redundancy in network traffic: Findings and implications," in Proc. SIGMETRICS, 2009, pp. 37–48.

15. 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. NSDI, 2010, pp. 28–28.

16. "PACK source code," 2011 [Online]. Available: http://www.eyalzo.com/ projects/ pack

17. A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: The implications of universal redundant traffic elimination," in Proc. SIGCOMM, 2008, pp. 219–0.

18. A. Anand, V. Sekar, and A. Akella, "SmartRE: An architecture for coordinated network-wide redundancy elimination," in Proc. SIGCOMM, 2009, vol. 39, pp. 87–98.

19. A. Gupta, A. Akella, S. Seshan, S. Shenker, and J. Wang, "Understanding and exploiting network traffic redundancy," UW- Madison, Madison, WI, USA, Tech. Rep. 1592, Apr. 2007.

20. M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global, cache local: YouTube network traffic at a campus network Measurements and implications," in Proc. MMCN, 2008, pp. 1–13.

21. S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing:Local algorithms for document fingerprinting," in Proc. SIGMOD, 2003, pp. 76–85.

22. S. Ihm, K. Park, and V. Pai, "Wide-area network acceleration for the developing world," in Proc. USENIX ATC, 2010

23. J.Srinivasan,W.Wei, X.Ma, andT. Yu, "EMFS:Esmail-based personal cloud storage," in *Proc. NAS*, 2011, pp. 248–257.

24. Amazon Elastic Compute Cloud (EC2). http://aws.amazon.com/ec2/.

25. "netfilter/iptables project: Libnetfilter_queue," Oct. 2005 [Online]. Available: http://www.netfilter.org/projects/libnetfilter_queu Author Dr.Sadhana Chidrawar Dean of MPGI COE Nanded Author Ankita Kotalwar ME(CSE) part II student

