International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 2, Issue 8, August 2015

Cost Reduction System using Prediction of Cloud Computing

[1]Ankita Kotalwar M.E(CSE) Student,
[2]Dr. Sadhana Chidrawar, Dean's MPGI College of Engineering Nanded

Abstract- **In this paper, we using cloud computing for eliminating traffic redundancy and reducing cost for a benefit of cloud customers. Here, we introduce the technique as bandwidth prediction through synchronization over user and server. The user's bandwidth were predicted by the server and proceed with the acknowledgement process, server of cloud provide the bandwidth to the end user of which customer needs. From this, cloud providing different bandwidth for different user which automatically eliminating end-to-end traffic redundancy and cost beneficial for an every customer accessing cloud. So easily cloud customer obtain their task and pay only for the usage in the cloud.**

Key words- **Cloud Computing, Bandwidth, Traffic redundancy Elimination**

I. INTRODUCTION

Cloud Computing is providing data, storage and resources to the customer use. Cloud is a large resources which provides to all over a scale. It task has enriched in this competitive world. Cloud providing a huge enhancement of data which are use or beneficial for the customers. Every customers are enrolled in the Cloud. Cloud has a various services as Infrastructure as a service [IaaS], Software as a Service [SaaS] and Platform as a Service [PaaS]. Each services gives various tools for using in system data. Cloud offer to a customer an economically pay-per-use method for their usage. Customers using cloud for their resources and data. Which are embedded in various techniques. Cloud customers are using cloud resources repeatedly and occur traffic due to accessing data, uploading and downloading file etc. Every user's accessing file in various place. So traffic occurring in the cloud server and providing services to the customers are reduced automatically i.e. the bandwidth given to the user has reduced and finally the cloud customer cannot able to access over in Cloud. Bandwidth provides are slow down due to traffic occur in server for getting more number for request from the user and given to the user which bandwidth is suitable for their process. According to that their bandwidth is predicted and provides to the cloud customers. This techniques has reduced the traffic between the users. The customer has pay their process only for the usage. The easily got pay for their only resources. For this process the traffic has reduced and enhanced the service to the cloud customer without any interrupt. So customer simultaneously use their data and resources.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. Therefore, it is commonly agreed that a universal, software-based, end-to-end TRE is crucial in today's pervasive environment [4,1].

1.1 Existing System

In the existing system the receiver based end-to-end TRE solution is used to eliminate the traffic redundancy by using predictions for future received chunks. In this solution, each receiver observes the incoming stream and tries to match its chunks with a previously received chunk chain or a chunk chain of a local file. Using the long- term chunks' meta-data information kept locally, the receiver sends to the server predictions that include chunks' signatures and easy-to-verify hints of the sender's future data. The sender first examines the hint and performs the TRE operation only on a hint- match. The purpose of this procedure is to avoid the expensive TRE computation at the sender side in the absence of traffic redundancy. When redundancy is detected, the sender then sends to the receiver only the ACKs to the predictions, instead of sending the data.

1.2 Disadvantages of Existing System

It does not provide authenticated users only to access the file and thus many users can access it, which raises the privacy issue and also load will be increased. The experimental results show that only a particular amount of traffic redundancy only eliminated by using this approach.

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.

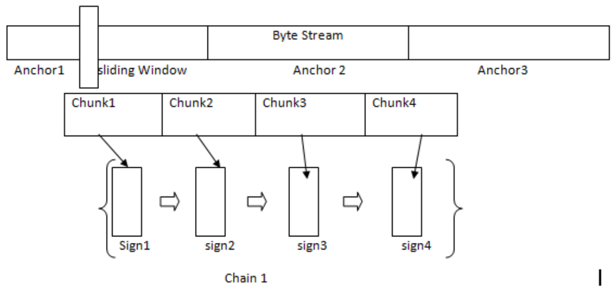


Fig1. From stream of Chain

III. THE PACK A LGORITHM

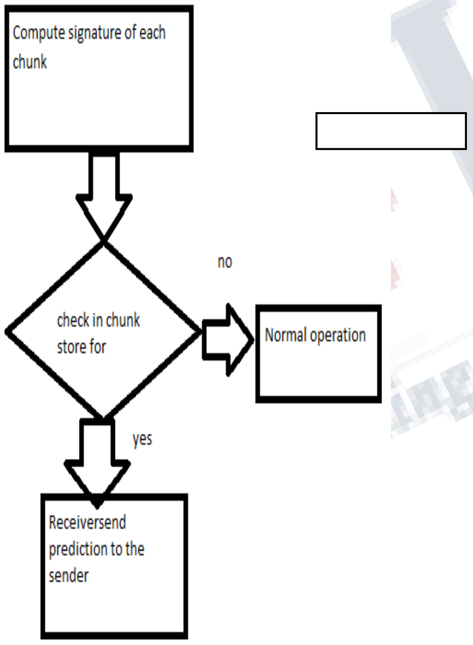
we first describe the basic receiver- driven operation of the PACK protocol. Several enhancements and optimizations are introduced. The stream of data received at the PACK receiver is parsed to a sequence of variable size, content-based signed chunks similar to [16][20]. The chunks are then compared 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

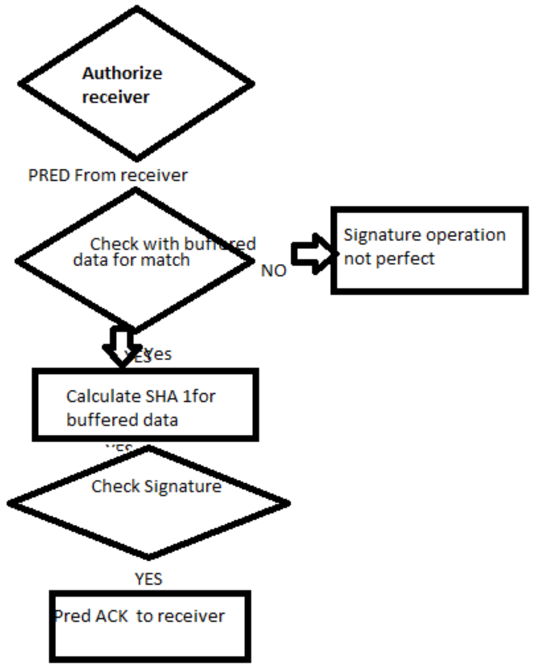
The arrival of a new information, the receiver side system that respective signature for every
Arrival of data from sender chunk and see the match in its local (temporary) chunk storage. If the chunk's match is founded, the receiver side determines whether it's a part of a formerly received chunk chain, 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.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.



Compute signature of each chunk
check in chunk store for
no
Normal operation
yes
Receiversend prediction to the sender

Receiver Side Algorithm

C) Sender Side Algorithm



Authorize receiver
PRED From receiver
Check with buffered data for match
NO
Signature operation not perfect
Yes
Calculate SHA 1for buffered data
Check Signature
YES
Pred ACK to receiver

Sender Side Algorithm

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) Hierarchical 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.

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.

5.1 Traffic Redundancy

5.1.1 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. 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

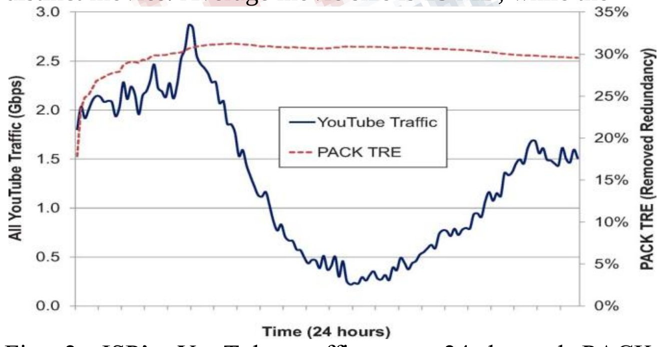


Fig. 2. ISP's YouTube traffic over 24 h, and PACK redundancy elimination ratio with this data.

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.

5.1.2 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.

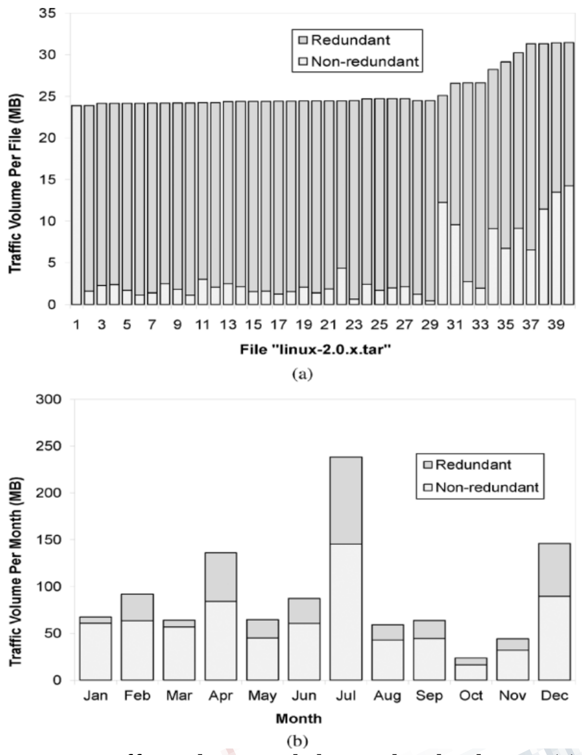


Fig. 3. Traffic volume and detected redundancy. (a) Linux source: 40 different
Linux kernel versions. (b) Email: 1-year Gmail account by month

VI. PREDICTION OPERATION

The chunks area unit predicting within the receiver, upon the arrival of recent knowledge the receiver computes the various signature for every chunk and appears for a match in its native chunk store. If the chunk's signature is found, the receiver determines whether or not it's a vicinity of a once received chain, victimisation the chunks' information. If affirmative, the receiver sends a prediction to the sender for many next expected chain chunks. Upon a thriving prediction, the sender responds with a PRED-ACK confirmation message. Once the PRED-ACK message is received and processed, the receiver copies the corresponding knowledge from the chunk store to its transmission control protocol input buffers, inserting it per the corresponding sequence numbers. At now, the receiver sends a traditional transmission control protocol ACK with ensuing expected transmission control protocol sequence variety.

6.1 Authentication

Consumer want to access their account means they have to sign in first with providing their personal user id and password. If users' given authentication will be valid means the user use their account. In this module mandatory and validation controls should be used.

6.2 Consumer Acknowledgement

In this module, acknowledgement process will be processed by cloud provider side. The predicted range ofconsumers' bandwidth has to be confirmed by consumer side. So predicted value has to be send to consumer and they will acknowledge for that means the costing will be calculate for consumer based on bandwidth.

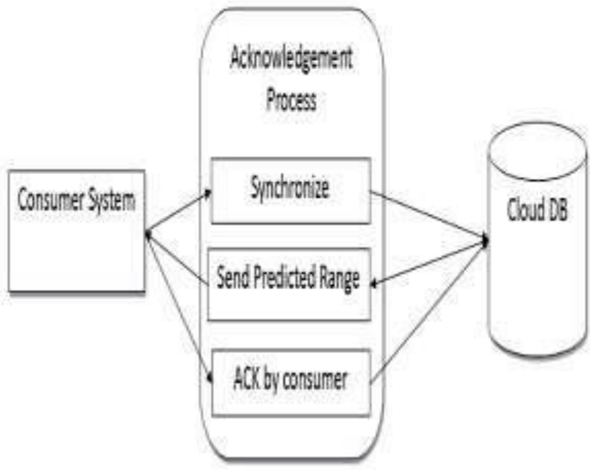


Fig4 Consumer Acknowledge

VII. IMPLEMENTATION

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.

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
```

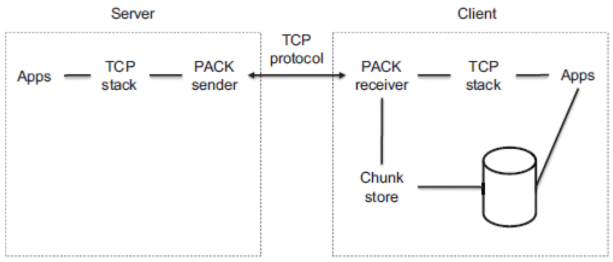


Fig5. Overview of PACK Implementation

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

$$D \leftarrow D\alpha + (1-\alpha)M$$

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

Pack Messages Format

In our implementation, we tend to use 2 presently unused TCP possibility codes,The first one is Associate in Nursing sanctionative possibility PACK permissible sent in an exceedingly SYN phase to point that the PACK possibility will be used when the connection is established. the opposite one could be a PACK message that will be sent over a longtime association once permission has been granted by each parties. 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.

Server Operational Cost

We measured the server performance and cost as a function of the data redundancy level in order to capture the effect of the TRAFFIC REDUNDANCY ELIMINATION mechanisms in real environment. To isolate the TRAFFIC REDUNDANCY ELIMINATION operational cost, we measured the server's traffic volume and CPU utilization at maximal throughput without operating a TRAFFIC REDUNDANCY ELIMINATION. We then used these numbers as a reference cost, based on present Amazon EC2 pricing. The server operational cost is com-posed of both the network traffic volume and the CPU utilization, as derived from the EC2 pricing.

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.

FUTURE WORK

Two interesting future extensions can provide additional benefits. First, our implementation maintains chains by keeping for any packet only the last observed sub-sequent packet in an LRU fashion. An interesting extension to this work is the statistical study of chains of packets that would enable multiple possibilities in both the packet 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 de-rived from receiver's power or server's cost changes.

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_queue.
[26]. Suresh Chougala et al. / International Journal of Computer Science & Engineering Technology (IJCSET) Survey on Traffic Redundancy and Elimination Approach for Reducing Cloud Bandwidth and Costs

1)Author Ankita Kotalwar ME(CSE) part II student
2)Author Dr.Sadhana Chidrawar Dean’s s MPGI COE Nanded

● ● ●