International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 9 Issue 3 –SEPTEMBER 2014.

# Pack: Prophecy-Based Technique for Eliminating Redundancy and Additional Cost

M. Sivananda[1], K. Sreekala[2]

[1] *M.Tech CSE, Sri Krishna Devaraya Engineering College, AP, INDIA*

[2] *Assistant Professor, CSE, Sri Krishna Devaraya Engineering College, AP, INDIA*

**Abstract— In this thesis, we explore Predictive ACKs (PACK), a new end-to-end TRE (traffic redundancy elimination) system, planned for cloud computing clients. Cloud-based traffic redundancy elimination wants to concern a sensible exploit of cloud resources then the price decrement bandwidth mutual with the extra price of traffic redundancy elimination calculation and preservation would be optimized. Predictive ACKs major benefit is its offloading potential of the cloud-server traffic redundancy elimination effort to end users, thus decreasing the processing expenditures encourages by the traffic redundancy elimination algorithm. Nothing comparable with existing solutions, Predictive ACKs won't need the server to constantly preserve users' status. This turns Predictive ACKs very appropriate for persistent calculation atmospheres that join server migration and client mobility to preserve cloud flexibility. Predictive ACKs is depended on a new traffic redundancy elimination technique, which permits the user to utilize chunks which are recently received to recognize chunk chains which are previously received, which in return can be utilized as dependable predictors to chunks which are future transmitted. We explore a complete functional Predictive ACKs execution, see-through to all Transmissions Control Protocol depended network devices and applications. At last, we evaluate Predictive ACKs advantages for cloud clients, utilizing traffic traces from different sources.**

## I. INTRODUCTION

Cloud computing presents its clients convenient and an economical pay-as-you-go service replica acknowledged. Cloud clients pay for only the actual utilize of computing assets, bandwidth, and preservation according to their altering requirements, using the cloud's elastic and scalable calculation abilities. Particularly information transfer expenditures (i.e., bandwidth) is an significant issue when trying to decrease expenses. Therefore, cloud clients, applying a sensible usage of the cloud's assets are motivated to utilize different traffic decrease techniques, particularly TRE (traffic redundancy elimination), for decreasing bandwidth expenditures. TRS (Traffic redundancy stems) from general end-users' actions, like frequently downloading, accessing, distributing, uploading (i.e., backup) and modifying/altering the similar or same data items (data, documents, video, and Web). Traffic redundancy elimination is utilized to remove the redundant content transmission and, consequently, to reduce the network price appreciably. In most general Traffic redundancy elimination solutions, both the receiver and sender observe and evaluate signatures of information chunks, parsed respective to the information content, proceeding to their transmission. When detecting redundant chunks, the sender restores the transmission of every redundant chunk with its tough signature. Commercial Traffic redundancy elimination solutions are admired at enterprise networks, and engage the deployment of couple or more proprietary-protocol, state synchronized middle-boxes at both branch offices and the intranet entry points of information centers and, removing recurring traffic among them.

While proprietary middle-boxes are popular point solutions within enterprises, they are not as attractive in a cloud environment. Cloud providers cannot benefit from a technology whose goal is to reduce customer bandwidth bills, and thus are not likely to invest in one. The rise of "on-demand" work spaces, meeting rooms, and work-from-home solutions [13] detaches the workers from their offices. In such a dynamic work environment, fixed-point solutions that require a client-side and a server-side middle-box pair become ineffective.

On the other hand, cloud-side elasticity motivates work distribution among servers and migration among data centers. Therefore, it is commonly agreed that a universal, software-based, end-to-end TRE is crucial in today's pervasive environment [14], [15]. This enables the use of a standard protocol stack and makes a TRE within end-to-end secured traffic (e.g., SSL) possible. Current end-to-end TRE solutions are sender-based. In the case where the cloud server is the sender, these solutions require that the server continuously maintain clients' status. We show here that cloud elasticity calls for a new TRE solution. 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 dueto lost synchronization. Second, the popularity of rich media that consume high bandwidth motivates content distribution

network (CDN) solutions, in which the service point for fixed and mobile users may change dynamically according to the relative service point locations and loads. Moreover, if an end-to-end solution is employed, its additional computational and storage costs at the cloud side should be weighed against its bandwidth saving gains.

## II. PROBLEM DEFINITION

In this thesis, we explored a novel end-to-end TRE solution which is receiver-based that relies on the predictions power to get rid of redundant traffic between the end-users and its cloud. In this resolution, every receiver monitors the incoming stream and attempts to match its chunks with a earlier received chunk chain or a local file chunk chain. With the help of the long-term chunks' metadata data kept locally, the receiver transmits to the server prophecies that contain chunks' signatures and simple-to- verify hints of the sender's future information. 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.

On the receiver side, we propose a new computationally lightweight chunking (fingerprinting) scheme termed PACK chunking. PACK chunking is a new alternative for Rabin fingerprinting traditionally used by RE applications. Experiments show that our approach can reach data processing speeds over 3 Gb/s, at least 20% faster than Rabin fingerprinting

.

## III. SYSTEM DEVELOPMENT

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. 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

Receiver Chunk Store

Receiver Algorithm Sender
Algorithm
Wire Protocol

Receiver Chunk Store:

PACK uses a new chains scheme. which chunks are linked to other chunks according to their last received order. The PACK receiver maintains a chunk store, which is a large size cache of chunks and their associated metadata. Chunk's metadata includes the chunk's signature and a (single) pointer to the successive chunk in the last received stream containing this chunk. When the new data are received and parsed to chunks, the receiver computes each chunk's signature using SHA-1. At this point, the chunk and its signature are added to the chunk store.

Receiver Algorithm

Upon a successful prediction, the sender responds with a PRED-ACK confirmation message. Once the PRED-ACK message is received and processed, the receiver copies the corresponding data from the chunk store to its TCP input buffers, placing it according to the corresponding sequence numbers. At this point, the receiver sends a normal TCP ACK with the next expected TCP sequence number. In case the prediction is false, or one or more predicted chunks are already sent, the sender continues with normal operation, e.g., sending the raw data, without sending a PRED-ACK message.

Sample Code:

```
<script type="text/javascript">
        function homePage(){

        document.commonForm.action='index_page.acti
on';
                document.commonForm.submit();
        }
        function regPage(){

        document.commonForm.action='register_page.a
ction';
                document.commonForm.submit();
        }
        function loginPage(){

        document.commonForm.action='login_page.acti
on';
                document.commonForm.submit();
        }
        function contactPage(){

        document.commonForm.action='contact_page.ac
tion';
                document.commonForm.submit();
```

```
        }
        function logoutPage(){

        document.commonForm.action='logout_page.act
ion';
                document.commonForm.submit();
        }
</script>
```

Sender Algorithm:

When a sender receives a PRED message from the receiver, it tries to match the received predictions to its buffered (yet to be sent) data. For each prediction, the sender determines the corresponding TCP sequence range and verifies the hint. Upon a hint match, the sender calculates the more computationally intensive SHA-1 signature for the predicted data range and compares the result to the signature received in the PRED message.

Wire protocol:

The existing firewalls and minimizes overheads; we use the TCP Options field to carry the PACK wire protocol. It is clear that PACK can also be implemented above the TCP level while using similar message types and control fields. The PACK wire protocol operates under the assumption that the data is redundant. First, both sides enable the PACK option during the initial TCP handshake by adding a PACK permitted to the TCP Options field. Then, the sender sends the (redundant) data in one or more TCP segments, and the receiver identifies that a currently received chunk is identical to a chunk in its chunk store. The receiver, in turn, triggers a TCP ACK message and includes the prediction in the packet's Options field. Last, the sender sends a confirmation message (PRED-ACK) replacing the actual data.

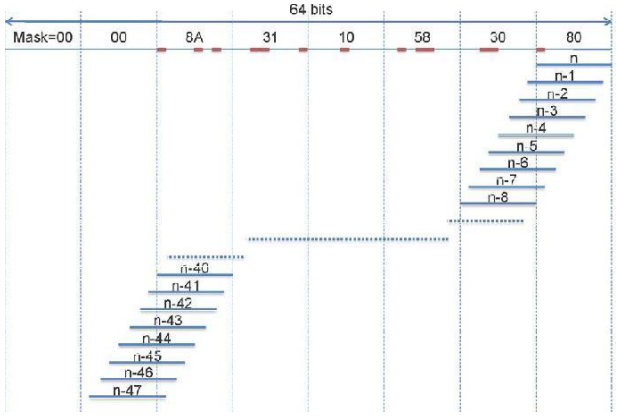



Figure1 PACK chunking: snapshot after at least 48 B were processed

Our measurements show that PACK chunking is faster than the fastest known Rabin fingerprint software implementation [31] due to one less XOR operation per byte. We further measured PACK chunking speed and compared it to other schemes. The measurements were performed on an unloaded CPU whose only operation was to chunk a 10-MB random binary file. Table V summaries the processing speed of the different chunking schemes. As a baseline figure, we measured the speed of SHA-1 signing and found that it reached 946 Mb/s.

PACK Impact on the Client CPU

To evaluate the CPU effort imposed by PACK on a client, we measured a random client under a scenario similar to the one used for measuring the server's cost, only this time the cloud server streamed videos at a rate of 9 Mb/s to each client. Such a speed throttling is very common in real-time video servers that aim to provide all clients with stable bandwidth for smooth view.

Chunking Scheme

Our implementation employs a novel computationally lightweight chunking (fingerprinting) scheme, termed PACK chunking. The scheme, presented in Proc. 8 and illustrated in Fig. 13, is an XOR-based rolling hash function, tailored for fast TRE chunking. Anchors are detected by the mask in line 1 that provides on average 8-kB chunks. The mask, as shown in Fig. 13, was chosen to consider all the 48 B in the sliding window

## IV. RELATED WORK

We measured the server performance and cost as a function of the data redundancy level in order to capture the effect of the TRE mechanisms in real environment. To isolate the TRE operational cost, we measured the server's traffic volume and CPU utilization at maximal throughput without operating a TRE. We then used these numbers as a reference cost, based on present Amazon EC2 [29] pricing. The server operational cost is composed of both the network traffic volume and the CPU utilization, as derived from the EC2 pricing. We constructed a system consisting of one server and seven clients over a 1-Gb/s network. The server was configured to provide a maximal throughput of 50 Mb/s per client. We then measured three different scenarios: a baseline no-TRE operation, PACK, and a sender-based TRE similar to EndRE's Chunk- Match [15], referred to as EndRE-like. For the EndRE-like case, we accounted for the SHA-1 calculated over the entire outgoing traffic, but did not account for the

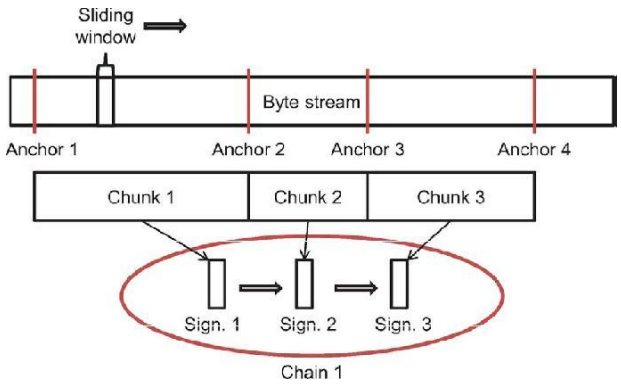
chunking effort. In the case of EndRE-like, we made the assumption of unlimited buffers at both the server and client sides to enable the same long-term redundancy level and TRE ratio of PACK.

References [17] and [18] 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 [19], [20], and [14]. In the latter, packet-level TRE techniques are compared [3], [21]. Our paper builds on their 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.

End RE [15] is a sender-based end-to-end TRE for enterprise networks. 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 B. 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 (around 10 MB per client), making the system inadequate formedium-to-large content or long-term redundancy. End RE 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.




## V. CONCLUSION

Cloud computing is anticipated to generate maximum demand for traffic redundancy elimination solutions as the quantity of information exchanged between its users and the cloud is estimated to significantly growth. The cloud architecture redefines the traffic redundancy elimination system necessities, building proprietary middle-box explanations insufficient. As a result, there is a growth of a TRE solution is required that decreases the cloud's functioning expenditure while accounting for cloud elasticity, user mobility, and application latencies. In this thesis, we have offered Predictive ACKs, a cloud-friendly, receiver-based, end-to-end traffic redundancy elimination that is depended on new speculative values that decreases latency and cloud functioning cost. Predictive ACKs don't need the server to constantly maintain users' status, thus enabling cloud elasticity and user mobility while preserving long-term redundancy. Moreover, PACK is capable of eliminating redundancy depending on content incoming to the user from multiple servers without applying a three-way handshake.

Our evaluation using awide 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.

## 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 and M.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 sub blocks," US Patent 5990810, Nov. 1999.\

[9] A. Flint, "The next workplace revolution," Nov. 2012 [Online].

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