International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Enhanced PACK Approach for Traffic Redundancy Elimination

Eldho Skaria[1], George T Vadakkumcheril[2]

[1,2]Department of Computer Science and Engineering, KMP College of Engineering, Asamannoor P.O Poomala, Odakkali, Kerala, India

Abstract: ***In this paper, we present PACK (Predictive ACKs), a novel end-to-end traffic redundancy elimination (TRE) system, designed for cloud computing clients. Cloud-based TRE needs to apply a careful use of cloud sources so that the bandwidth cost decrease along with the additional price of TRE computation and storage space would be enhanced. PACK's main advantage is its ability of offloading the cloud-server TRE effort to end clients, thus reducing the computing costs caused by the TRE algorithm. PACK is depending on a novel TRE strategy, which allows the client to use recently obtained chunks to recognize formerly received chunk chains, which in turn can be used as efficient predictors to future transmitted chunks. We existing a completely efficient PACK implementation, transparent to all TCP-based programs and network devices. Lastly, we evaluate PACK benefits for cloud clients.***

Keywords: Traffic Redundancy Elimination, Cloud computing, Chunks, Predictive Acknowledgement.

1. Introduction

Cloud computing provides its clients an economical and practical pay-as-you-go service design, known also as usage-based costs [1]. Reasoning customers pay only for the real use of computing sources, storage space, and bandwidth usage, according to their modifying needs, using the cloud's scalable and flexible computational abilities. In particular, data exchange expenses (i.e., bandwidth) is an important issue when trying to reduce expenses [1]. Consequently, cloud clients, applying a careful use of the cloud's sources, are inspired to use various traffic decrease methods, in particular traffic redundancy elimination (TRE), for reducing bandwidth costs.

Traffic redundancy arises from typical end-users' actions, such as continuously accessing, downloading, uploading (i.e., backup), circulating, and changing the same or identical information items (documents, details, Web, and video). TRE is used to remove the transmitting of recurring material and, therefore, to considerably decrease the network cost In most common TRE alternatives, both the sender and the recipient analyze and compare signatures of details chunks, parsed according to the data content, before their transmitting. When recurring chunks are detected, the sender changes the transmitting of each redundant chunk with its powerful signature [2] [3].

We show here that cloud flexibility demands a new TRE remedy. First, cloud fill controlling and power optimizations may lead to a server-side procedure and information migration atmosphere, in which TRE alternatives that require full synchronization between the server and the consumer are hard to achieve or may reduce efficiency due to missing synchronization. Second, the reputation of rich media that eat high information transfer usage encourages content distribution network (CDN) alternatives, in which the support point for set and cellular customers may modify dynamically according to the comparative support factor places and plenty. Moreover, if an end-to-end solution is employed, its additional computational and storage space costs at the cloud side should be compared to its bandwidth saving gains.

In this paper, I present a novel receiver-based end-to-end TRE remedy that depends on the energy of forecasts to eliminate redundant traffic between the cloud and its end-users. In this remedy, each recipient notices the inbound flow and tries to match its chunks with a formerly obtained amount sequence or a chunk chain of a local file. Using the long-term chunks' metadata information kept locally, the recipient delivers to the server predictions including chunks' signatures and easy-to-verify hints of the sender's upcoming details. The sender first investigates the hint and works the TRE function only on a hint-match. The purpose of this process is to prevent the costly TRE computation at the sender part in the lack of traffic redundancy. When redundancy is recognized, the sender then delivers to the receiver only the ACKs to the predictions, instead of delivering the data.

2. Related Work

There are two interesting related works available. The first one is Wanax and the second one is EndRE.

2.1 Wanax

Wanax [4] is a TRE program for the developing world where storage and WAN data transfer usage are limited. It is a software-based middle-box replacement for the expensive commercial hardware. In this plan, the sender middle-box keeps returning the TCP stream and sends data signatures to the receiver middle-box. The receiver examine whether the data is found in its local cache. Data chunks that are not discovered in the storage space cache are fetched from the middle-box or a nearby receiver middle-box. Naturally, such a scheme incurs three-way-hand shake latency for non cached data.

2.2 EndRE

EndRE [5] is a sender-based end-to-end TRE for enterprise networks. It uses a new chunking plan that is quicker than the commonly used Rabin fingerprint, but is limited to sections as small as 32–64 B. Compared with PACK, EndRE needs

the server to maintain a completely and effectively synchronized storage cache for each customer. To conform with the server's storage specifications, these caches are kept little (around 10 MB per client), creating the program in adequate for medium-to-large material or long-term redundancy. EndRE is server-specific, hence not appropriate for a CDN or cloud environment

3. PACK Algorithm

The flow of information obtained at the PACK recipient is parsed to a series of variable-size. The chunks are then compared to the receiver local storage space, known as chunks store. If a related chunks is found in the local chunks store, the recipient retrieves the sequence of following chunks, generally known as a chain, by traversing the series of LRU chunks suggestions that are included in the chunks' meta-data. Using the designed sequence, the receiver sends a predication to the sender for the following information. Part of each chunk's predication, known as 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 expected information or a byte-wide XOR checksum of all or chosen bytes. The predication sent by the receiver contains the variety of the expected information, the sign, and the signature of the chunks. The sender recognizes the predicted range in its buffered information and confirms the sign for that variety. If the result matches the obtained hint, it is constantly on the perform the more computationally intense SHA-1 signature function. Upon a signature match, the sender delivers a verification message to the recipient, allowing it to duplicate the printed information from its local storage space.

3.1 Receiver Chunk Store

PACK uses a new chains plan, described in Fig. 1, in which chunks are connected to other chunks according to their last received order. The PACK recipient preserves a chunks store, which is a huge dimension storage cache of chunks and their associated metadata. Chunk's meta-data contains the chunk's signature and a (single) suggestion to the subsequent amount in the last received stream containing this amount. Caching and listing techniques are applied to effectively sustain and recover the stored chunks, their signatures, and the chain established by traversing the chunk suggestions.

When the new information are obtained and parsed to chunks, the receiver determines each chunk's signature using SHA-1. At this point, the chunks and its signature are included to the chunks store. In inclusion, the meta-data of the formerly obtained chunks in the same flow is modified to factor to the present chunk.

The unsynchronized characteristics of PACK allows the recipient to map each current file in the local file system to a sequence of chunks, preserving in the chunks store only the meta-data associated with the chunks

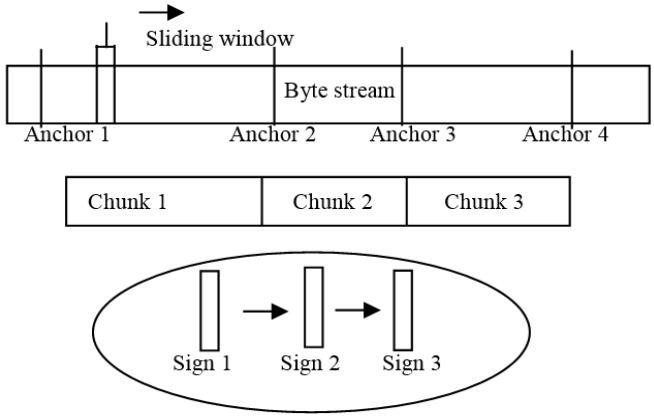

Figure 1: From stream to chain.

The employment of a small chunk dimension provides better redundancy elimination when information variations are fine-grained, such as infrequent changes in an HTML page. On the other hand, the use of smaller chunks improves the storage catalog dimension, memory utilization, and permanent magnetic hard drive seeks.

3.2 Receiver Algorithm

Upon the appearance of new information, the recipient determines the respective signature for each and looks for a coordinate in its local chunks store. If the chunk's signature is discovered, the receiver determines whether it is a aspect of a formerly obtained sequence, using the chunks' meta-data. If positive, the recipient delivers a prediction to the sender for several next predicted sequence sections. The predictions has a place to start in the byte flow (i.e. offset) and the identification of several following chunks (PRED command).

Upon a effective prediction, the sender reacts with a PRED-ACK verification concept. Once the PRED-ACK message is obtained and prepared, the recipient duplicates the corresponding data from the chunks store to its TCP feedback buffers, placing it according to the corresponding series numbers. At this factor, the recipient delivers a regular TCP ACK with the next predicted TCP series variety. In situation the prediction is incorrect, or one or more predicted sections are already sent, the sender carries on with regular function

3.3 Sender Algorithm

When a sender receives a PRED message from the receiver, it tries to match the received predictions to its buffered 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. Note that in case the hint does not match, a computationally expansive operation is saved. If the two SHA-1 signatures match, the sender can safely assume that the receiver's prediction is correct. In this case, it replaces the corresponding outgoing buffered data with a PRED-ACK message

3.4 Wire Protocol

To be able to adjust with current firewalls and minimize overheads, we use the TCP Choices area to bring the PACK wire method. It is obvious that PACK can also be implemented above the TCP stage while using identical concept kinds and control areas.

Fig. 2 demonstrates the way the PACK wire method operates under the supposition that the information is repetitive. First, both sides enable the PACK choice during the preliminary TCP handshake by including a PACK allowed banner (denoted by a strong line) to the TCP Choices area. Then, the sender delivers the (redundant) data in one or more TCP chunks, and the recipient recognizes that a currently obtained chunks is similar to a chunks in its chunk store. The recipient, in convert, activates a TCP ACK concept and includes the predictions in the packet's Choices area. Last, the sender delivers a verification message (PRED-ACK) replacing the real information.

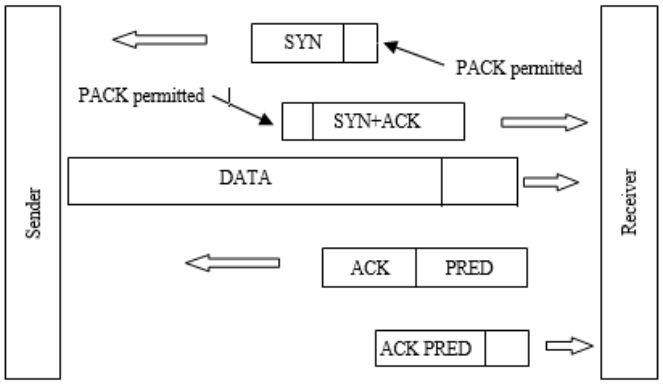

Figure 2: PACK wire protocol

4. Optimizations

In this section we discuss with additional options of PACK system. Here we describe two additional options. The first one is Adaptive receiver virtual window and the second one is Cloud server as a receiver

4.1 Adaptive Receiver Virtual Window

PACK allows the recipient to locally acquire the sender's data when a local duplicate is available, thus removing the need to send this information through the system. We phrase the receiver's fetching of such local information as the reception of virtual data.

When the sender sends a higher number of virtual data, the connection amount may be, to a certain level, restricted by the number of predictions sent by the recipient. This, in convert, means that the recipient predictions and the sender confirmation should be fast to be able to achieve great virtual data amount.

PACK allows the recipient to merge several chunks into a single range, as the sender is not surrounded to the anchor bolts originally used by the receiver's information chunking criteria. The combined range has a new hint and a new signature that is an SHA-1 of the concatenated content of the chunks.

The varying predictions size presents the idea of a virtual window, which is the present receiver's screen for virtual data. The virtual window is the receiver's higher limited for the aggregated variety of bytes in all the awaiting predictions. The virtual window is first set to a little value, which is identical to the receiver's flow control window. The recipient increases the virtual window with each predictions achievements

Upon the first chunks coordinate, the recipient delivers predictions limited to its preliminary virtual window. It is likely that, before the predictions reach the sender, some of the corresponding real data is already passed on from it. When the actual information comes, the recipient can partly validate its predictions and improve the virtual screen. Upon getting PRED-ACK confirmations from the sender, the recipient also improves the virtual window. This logic appears like the slow-start aspect of the TCP rate control algorithm. When a mismatch happens, the recipient changes returning to the preliminary virtual window.

4.2 Cloud Server as a Receiver

In a growing pattern, reasoning storage space is becoming a dominant player [6] from back-up and discussing solutions [7] [10] to the American Nationwide Collection, and e-mail solutions [8] [9]. In many of these solutions, the reasoning is often the recipient of the data.

If the delivering customer has no energy restrictions, PACK can work to preserve data transfer usage on the upstream to the cloud. In these situations, the end-user functions as a sender, and the cloud server is the recipient. The PACK algorithm need not modify. It does require, however, that the cloud server like any PACK receiver maintain a chunk store.

5. Enhanced PACK Approach

An interesting extension to this work is to enable multiple possibilities in both the chunk order and the corresponding predictions. The proposed 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.

In the enhanced PACK, the receiver node stores the chunk value into the chunk list for a whole data transmission. It will mark the end of each and every data transmission. It will do the following process. When it find the match in the chunk store, it will send the prediction message not only for the single data, it will send the prediction message by include all the messages until the end mark. If the prediction of receiver is correct, the sender checks the signature and sends the PRED_ACK for the prediction message to the receiver.

- Split the user entered data into words
- Convert each word to hexadecimal
- That hexadecimal value is taken as the chunk for the data entered by the user.

- The chunk value is put in the chunk list.
- Add Signature to each chunk
- Transmit the data with the chunk value.

After the arrival of data from sender, the receiver checks the received chunk value with the chunk store. If the chunk value is already in the chunk store, it will send the prediction to the sender node with the signature. The sender will verify the prediction message from the receiver in the following manner

- Check the prediction message with the buffered data
- If matches, calculate SHA-1for buffered data
- The sender compares the received signature with the calculated SHA-1 value. If it matches then the sender node send the PRED_ACK to the receiver

The main advantage of EPACK is instead of checking the chunk value for each and every word it receives, it can do multiple word prediction at the same time.

6. Performance Evaluation

In this section, we demonstrate the performance of the proposed public auditing scheme using Network Simulator 2. We are interested in the following performance metrics: Packet Delivery Ratio, Packet Loss Ratio, End-to-End Delay

6.1 Packet Delivery Ratio

The ratio of packets that are successfully delivered to a destination compared to the number of packets that have been sent out by the sender. It demonstrates the amount of data delivered to the destination.

Packet Delivery Ratio = ∑ Number of packets received / ∑ Number of packets send.

Higher value of packet delivery ratio signifies better performance of the authentication scheme.

6.2 Packet Loss Ratio

The total amount of packets losses during the simulation is said to be the packet loss ratio.

Packets lost = Number of packets send – Number of packets received.

Lower value of packet loss ratio signifies better performance of the authentication scheme

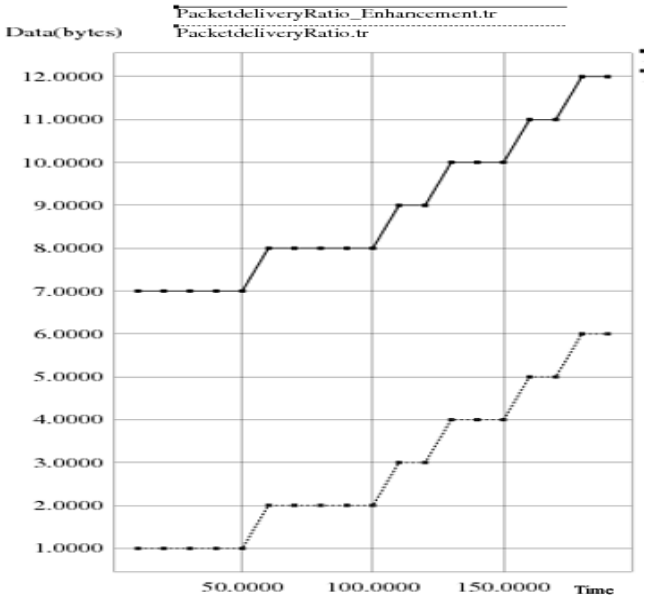

Figure 3: Packet Delivery Ratio

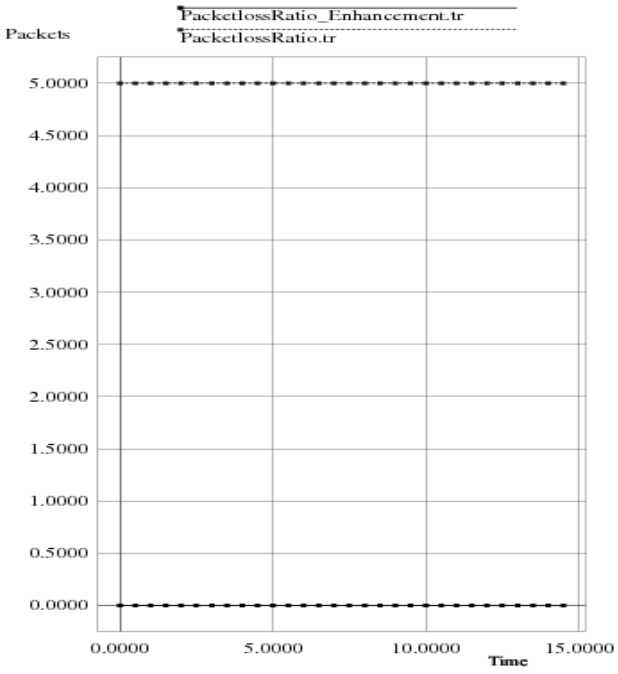

Figure 4: Packet Loss Ratio

6.3 End-to-End Delay

The time taken in average by a data packet for arriving at its intended destination.

End-to-End delay = ∑ (arrival time – sending time) / ∑ Number of connections.

Lower value of end-to-end delay signifies better performance of the protocol.

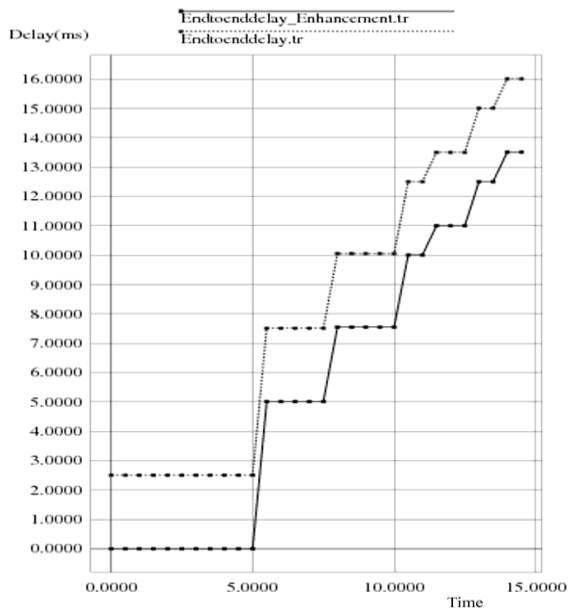

Figure 5: End-to-End Delay

7. Conclusion

Cloud computing is predicted to trigger high demand for TRE solutions as the amount of data exchanged between the cloud and its customers is expected to considerably increases. In this work, we have provided PACK, a receiver-based, cloud-friendly, end-to-end TRE that is depending on novel speculative principles that decrease latency and cloud operational cost. PACK does not need the server to consistently maintain clients' position, thus allowing cloud flexibility and customer mobility while protecting long-term redundancy. Moreover, PACK is capable of removing redundancy depending on material coming to the customer from several web servers without implementing a three-way handshake.

References

[1] 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.

[2] U. Manber, "Finding similar files in a large file system," in Proc. USENIX Winter Tech. Conf., 1994, pp. 1–10.

[3] A Muthitacharoen, B. Chen, and D. Mazières, "A low-bandwidth network file system," in Proc. SOSP, 2001, pp. 174–187.

[4] S. Ihm, K. Park, and V. Pai, "Wide-area network acceleration for the developing world," in Proc. USENIX ATC, 2010, pp. 18–18.

[5] 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.

[6] H. Stevens and C. Pettey, "Gartner says cloud computing will be as influential as e-business," Gartner Newsroom, Jun. 26, 2008.

[7] "Dropbox," 2007 [Online]. Available: http://www.dropbox.com/

[8] D. Hansen, "GMail filesystem over FUSE," [Online]. Available: http:// sr71.net/projects/gmailfs/

[9] J.Srinivasan,W.Wei, X.Ma, andT. Yu, "EMFS:Email-based personal cloud storage," in Proc. NAS, 2011, pp. 248–257.

[10] H. Abu-Libdeh, L. Princehouse, and H.Weatherspoon, "RACS: A casefor cloud storage diversity," in *Proc. SOCC*, 2010, pp. 229–240.

Author Profile

Eldho Skaria received the B.Tech Degree in Information Technology from Anna University Chennai, Tamil Nadu, India, in 2011. He is currently pursuing M.Tech Degree in Computer Science and Engineering with Specialization in Cyber Security from Mahatma Gandhi University, Kerala, India.

George T. Vadakkumcheril received degree of M.Tech Computer Science & Engineering from Karunya University, Coimbatore in 2014 and B.Tech in Information Technology from Mahatma Gandhi University, Kerala in 2011. He is working as Assistant Professor in Computer Science & Engineering Department under Mahatma Gandhi University, Kerala.