- ***Receiver/Sender Algorithm***

Upon the arrival of recent information, the receiver computes the several 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 region of a once received chain, exploitation the chunks' information. If affirmative, the receiver sends a prediction to the sender for many next expected chain chunks. The prediction carries a start line within the computer memory unit stream (i.e., offset) and also the identity of many resulting chunks (PRED command). Upon a booming prediction, the sender responds with a PRED-ACK confirmation message.

- ***Sender/Receiver 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) knowledge. for every prediction, the sender determines the corresponding TCP sequence vary and verifies the hint. Upon a int match, the sender calculates the additional computationally intensive SHA-1 signature for the anticipated knowledge vary and compares the result to the signature received within the PRED message.

- ***Wire Protocol***

In order to conform with existing firewalls and minimize overheads, we use the TCP Options field to carry the HYPACK wire protocol. It is clear that

## 3. End-Redundancy Elimination TRE

EndRE [5] end-system redundancy elimination provides quick, reconciling and penurious in memory usage so as to opportunistically leverage resources on finish hosts. EndRE is predicated on 2 modules server and therefore the shopper. The server-side module is accountable for distinguishing redundancy in network knowledge by comparison against a cache of previous knowledge and cryptography the redundant knowledge with shorter meta-data. The client-side module consists of a hard and fast size circular FIFO log of packets and easy logic to rewrite the meta-data by "de-referencing" the offsets sent by the server. Thus, most of the quality in EndRE is especially on the server aspect. so it's server specific ineffectual to keep up the full synchronization between shopper and therefore the server. EndRE uses Sample computer memory unit process theme that is faster than Rabin process. EndRE restricted for tiny redundant chunks of the order of 32-64 bytes. solely distinctive chunks ar transmitted between file servers and purchasers, leading to lower information measure consumption. the essential plan underlying EndRE is that of content-based naming wherever associate degree object is divided into chunks and indexed by computing hashes over chunks.

Comparison with Novel-TRE:

1) It is server specific
2) Chunk size is small

## 4. Novel TRE

The novel-TRE approach depends on the ability of predictions to eliminate redundant traffic between its endusers and therefore the cloud. during this technique, every receiver observes the incoming stream and tries to match its chunks with a antecedently received chunk chain or a piece chain of a neighborhood file. mistreatment the semipermanent chunks' data info unbroken regionally, the receiver sends to the server predictions that embrace chunks' signatures and easy-to verify hints of the sender's future knowledge. On the receiver facet, we tend to propose a brand new computationally light-weight constellation [1] (fingerprinting) theme. light-weight constellation is various for Rabin procedure [8] historically utilized by RE applications with high processing speed.

### A. Chunking Mechanism

Novel-TRE uses a replacement chains theme within which chunks ar connected to alternative chunks in line with their last received order. The novel-TRE receiver maintains a piece store, that could be a massive size cache of chunks and their associated information. Chunk's information includes the chunk's signature and a (single) pointer to the sequent chunk within the last received stream containing this chunk. once the new information ar received and parsed to chunks, the receiver computes every chunk's signature exploitation SHA-1.

**Proc. 1:** Sender/Receiver Segment Processing
1. **if** segment carries payload *data* **then**
2. calculate chunk
3. **if** reached chunk boundary **then**
4. activate predAttempt()
5. **end if**
6. **else if** PRED-ACK segment **then**
7. processPredAck()
8. activate predAttempt()
9. **end if**

### B. Prediction Operation:

The chunks square measure predicting within the receiver, upon the arrival of latest 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 section of a at one time received chain, victimization the chunks' information.If affirmative, the receiver sends a prediction to the sender for many next expected chain chunks.Upon a winning 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 communications protocol input buffers, inserting it in line with the corresponding sequence numbers. At this time, the receiver sends a traditional communications protocol ACK with successive expected communications protocol sequence range..

**Proc. 2:** predAttempt()
1. **if** received *chunk* matches one in chunk store **then**
2. **if** foundChain(*chunk*) **then**
3. prepare PREDs

4. send single TCP ACK with PREDs according to Options free space
5. exit
6. **end if**
7. **else**
8. store *chunk*
9. link *chunk* to current chain
10. **end if**
11. send TCP ACK only

### C. HyPACK Messages Format

In our implementation, we use two currently unused TCP option codes, similar to the ones defined in SACK [16]. The first one is an enabling option HYPACK permitted sent in a SYN segment to indicate that the HYPACK option can be used after theconnection is established. The other one is a HYPACK message that may be sent over an established connection once permission has been granted by both parties.

**Proc. 3:** processPredAck()
1. **for all** offset PRED-ACK **do**
2. read data from chunk store
3. put data in TCP input buffer
4. **end for**

**Proc.4:processLruClr()**
**1.for all received chunk with timestamp**
2.if new=exist then
3.Update timestamp
4.Else
5.If buffer not full then
6.Add new entry with new timestamp
7.Else
8.Search for larger timestamp
9.And replace with new entry
10.End if
11.End if

## 5. 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 HYPACK 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.

### A. 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 HyPACKet 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.

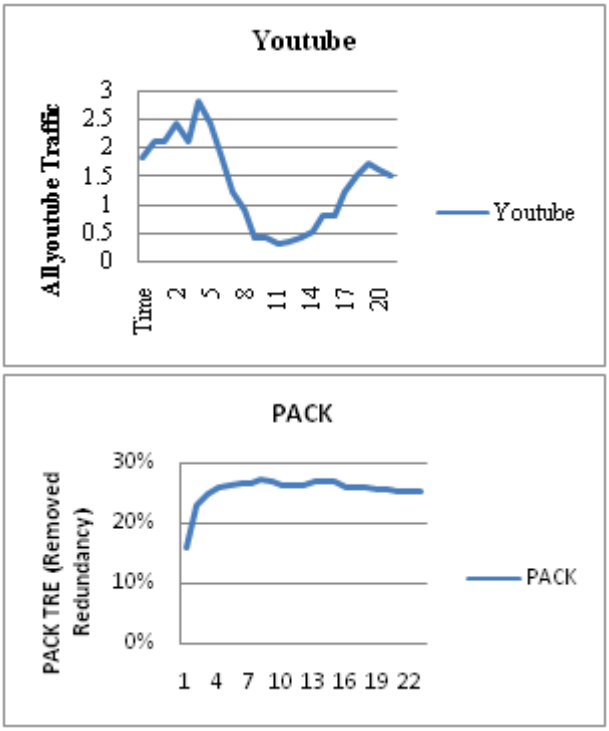


**Figure 1:** ISP's YouTube traffic over 24 h, and HYPACK redundancy elimination ratio with this data.

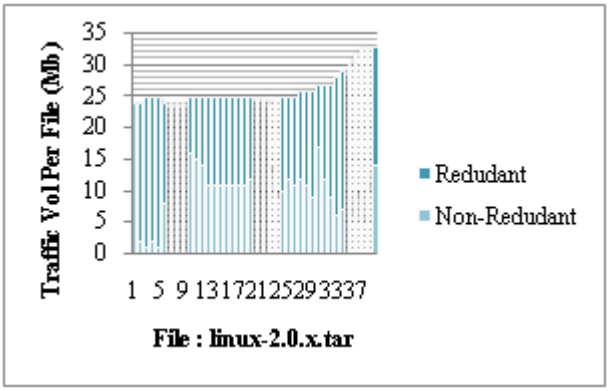


**Figure 2:** Traffic volume and detected redundancy. (a) Linux source: 40 different Linux kernel . versions.

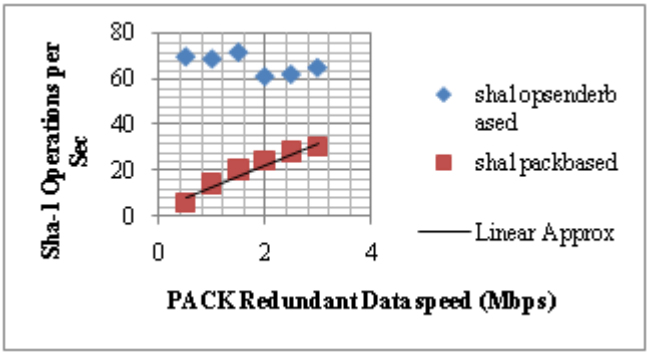


**Figure 3:** Difference in computation efforts between receiver and sender-driven modes for the transmission of Email data collection. (a) Server effort as a function of time. (b) Sender effort relative to redundant chunks signatures download time (virtual speed).

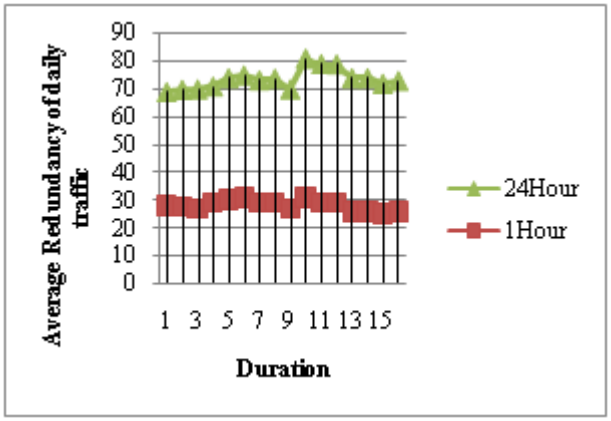


**Figure 4:** Social network site: traffic redundancy per day with different time lengths of cache

## 6. Conclusion

we planned a novel-TRE approach for eliminating redundancy within the cloud atmosphere. Our planned theme has vital options like reduces the transmission value by predicting chunks, redundancy detection by the shopper, doesn't need the server to unceasingly maintain clients' standing. Our receiver and sender based mostly end-to-end TRE suites for cloud atmosphere. Cloud computing is predicted to trigger high demand for TRE solutions because the quantity of information changed between the cloud and its users is predicted to dramatically increase. The cloud atmosphere redefines the TRE system necessities, creating proprietary middle-box solutions inadequate. Consequently, there's a rising would like for a TRE resolution that reduces the cloud's operational value atleast 30% but before whereas accounting for application latencies, user quality, and cloud physical property.

## References

[1] "PACK:PreductionBaed cloud bamdwidth and cost reduction system" Eyal Zohar, Israel Cidon, and Osnat Mokryn IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 22, NO. 1, FEBRUARY 2014
[2] Impact of Cloud Computing Virtualization Strategies on Workloads' Performance.
[3] 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
[4] 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
[5] U. Manber, "Finding similar files in a large file system," in Proc. USENIX Winter Tech. Conf., 1994, pp. 1–10.
[6] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in Proc. SIGCOMM, 2000, vol.30, pp. 87–95.
[7] A. Muthitacharoen, B. Chen, and D. Mazières, "A low-bandwidth network file system," in Proc. SOSP, 2001, pp. 174–187.
[8] 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.
[9] 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.
[10] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, "Redundancy in network traffic: Findings and implications," in Proc. SIGMETRICS, 2009, pp. 37–48.
[11] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "HyPACKet caches on routers: The implications of universal redundant traffic elimination," in Proc. SIGCOMM, 2008, pp. 219–230.
[12] 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.
[13] 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.
[14] 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.
[15] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local algorithms for document fingerprinting," in Proc. SIGMOD, 2003, pp.76–85.
[16] S. Ihm, K. Park, and V. Pai, "Wide-area network acceleration for the developing world," in Proc. USENIX ATC, 2010, pp. 18–18.
[17] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," RFC 2018, 1996.