RESEARCH ARTICLE OPEN ACCESS

Traffic Redundancy Elimination and Bandwidth Optimization over Cloud Using Cooperative End-to-End Tre Solution Technique

Mr.Y Srinivasa Rao[1], Dasari Naga Ratna Sirisha[2]

*[1] Assistant Professor, Department of MCA, Vignan's Lara Institute of Technology & Science, Vadlamudi, Guntur, Andhra Pradesh, India

[2]MCA Student , Department of MCA, Vignan's Lara Institute of Technology & Science, Vadlamudi, Guntur, Andhra Pradesh, India

Abstract:

The compensation as-you-go benefit show affects cloud clients to diminish the utilization cost of bandwidth. Traffic Redundancy Elimination (TRE) has been appeared to be a compelling answer for lessening bandwidth costs, and in this manner has as of late caught huge consideration in the cloud environment. By concentrate the TRE procedures in a follow driven approach, we found that both here and now (time traverse of seconds) and long haul (time traverse of hours or days) information redundancy can simultaneously show up in the traffic, and exclusively utilizing either sender-based TRE or receiver-based TRE can't all the while catch the two sorts of traffic redundancy. Additionally, the productivity of existing receiver-based TRE arrangement is powerless to the information changes contrasted with the chronicled information in the reserve. In this paper, we propose a Cooperative end-to-end TRE arrangement (CoRE) that can identify and evacuate both here and now and long haul redundancy through a two-layer TRE outline with agreeable activities between layers. A versatile forecast calculation is additionally proposed to enhance TRE effectiveness through powerfully modifying the expectation window measure based on the hit proportion of chronicled expectations. Plus, we upgrade CoRE to adjust to various traffic redundancy attributes of cloud applications to enhance its activity cost. Broad assessment with a few genuine follows demonstrate that CoRE is prepared to do viably recognizing both here and now and long haul redundancy with low extra cost while guaranteeing TRE proficiency from information changes.

Keywords **—** Traffic redundancy elimination, cloud computing, network bandwidth, bandwidth cost, end-to-end

1 INTRODUCTION

CLOUD registering is a developing IT worldview that expert vides utility figuring by a compensation as-you-go benefit display [1]. An ever increasing number of associations are moving their organizations to the cloud to give administrations like video streaming, Web and person to person communication. With present day virtu-alization advancements, the cloud gives asset flexible ity [2], which empowers the limit of cloud applications to scale here and there on request to adjust to the adjustments in workloads. In any case, the organization of web and video benefits in cloud drive expanding departure bandwidth requests because of information access from a lot of clients. An outstanding case is Netflix [3], which has facilitated its video-streaming administration in the cloud with Amazon Web Services [4] since 2009. The cost of cloud facilitating administrations can boundlessly increment because of the utilization based bandwidth pric-ing. Amazon EC2 charges the information exchange out to Internet can be $0.155 per GB for initial 10 TB/month [4], and nowa-days it is simple for a cloud server to have 1TB month to month datatransfer. In this way, the bandwidth cost has turned into a genuine worry for the cloud application organization and got a ton of consideration [5], [6].

With a specific end goal to diminish the bandwidth cost for information exchange from the cloud, Traffic Redundancy Elimination (TRE) tech-nologies have being abused [7] to lessen the bandwidth use by dispensing with the transmission of copy informa-tion. In conventional TRE arrangements [8], [9], the sender distinguishes the copy content by contrasting the active information and its neighborhood reserve which stores already transmitted bundles, and after that sends the reference of copy information to the receiver rather than crude information. The receiver gets the information from its nearby store by the reference query. These techniques need to keep up completely synchronized reserves at both the sender and receiver with a specific end goal to store already transmit-ted/got bundle payload. In any case, the workload dis-tribution and movement for versatility in the cloud can prompt regular changes of administration focuses for the customers, which makes reserve synchronization troublesome and may bring about corrupted TRE effectiveness. In this manner, customary TRE arrangements are illsuited to the cloud environment, as noted in [7]. Moreover, considering the use based asset valuing of the cloud, the use of calculation, stockpiling and bandwidth assets for running TRE software may kill the band-width cost investment funds got by TRE. Without cloud flexible ity, the heap brought about by TRE at the servers may adversely influence the execution of the cloud applications. With the flexible limit scaling in cloud, the overhead of TRE may trigger the scaling up of the application's ability through including more servers, which expands the activity cost of the application. Therefore, guaranteeing low asset cost is nec-essary for TRE in the cloud environment.

To address the above issues of TRE in cloud, a receiver-based TRE arrangement named PACK [7], [10] has been ace postured. In PACK, once a customer gets an information lump that as of now exists in its neighborhood store, it is normal that the future approaching information would likewise be copy. In this manner, the customer pre-dicts the future coming information pieces and advises the cloud

server with the marks of the anticipated lumps. The server thinks about the anticipated marks got from the customer with the marks of active lumps and affirms the effectively anticipated pieces, which at that point don't should be exchanged. Without keeping up the customers' status at the server, PACK successfully handles the cloud flexibility. By foreseeing future traffic redundancy at customers, PACK off-loads most calculation and capacity cost to customers, and along these lines enormously diminishes the TRE cost in cloud servers.

Then again, past investigations on arrange traffic redundancy [11] demonstrate two kinds of redundancy, alluded to as here and now traffic redundancy (reiteration in minutes) and long haul traffic (redundancy in hours or days) as indicated by the time size of reiteration event. It is discovered that dominant part of information coordinates between the traffic and the store happen for information under 150 bytes and have high level of worldly territory (e.g., 60 percent inside 100 seconds), while well known lumps can repeat with a period vary ence as extensive as 24 hours [11]. By concentrate genuine follows, we show that the here and now and long haul redundancy may exist together in the system traffic, and PACK can effec-tively catch long haul redundancy in the traffic between a server and a customer yet neglects to catch here and now redun-dancy. Since the customers can reserve a lot of his-torical parcel information on huge size constant stockpiles (e.g., circles) in the long haul, PACK can wipe out the traffic content rehashed in a long stretch with the customers' traffic redundancy forecasts. Nonetheless, on the grounds that PACK matches information pieces at normal size of 8 KB, it can't distinguish the fleeting redundancy that shows up at fine-granularity (e.g., the information estimate around 150 bytes). Since an expansive segment of redundancy is found in here and now time scale, PACK can't abuse the full redundancy in the system traffic. In any case, utilizing a little piece estimate in PACK isn't an adequate solu-tion to the issue, since generally the bandwidth cost for transmitting lump expectations would destroy the band-width funds from wiping out piece transmissions.

In this paper, we intend to plan an effective TRE answer for administrations that are sent in the cloud and ruled by the information exchange from the cloud servers to the customers. We propose a Cooperative end-to-end TRE arrangement, named as CoRE, with ability of expelling both here and now and long haul redundancy to such an extent that traffic redundancy can be disposed of to the most noteworthy degree. Center includes two layers of helpful TRE activities. The main layer TRE performs forecast based Chunk-Match like PACK [7] to catch long haul traffic redundancy. The second-layer TRE identi-fies maximal copy substrings inside an active piece contrasted and the beforehand transmitted lumps in a nearby piece store at the sender, alluded to as In-Chunk Max-Match. On the off chance that the redundancy location at the main layer falls flat, CoRE swings to the second-layer to recognize better granularity redundancy, i.e., here and now redundancy, inside pieces.

With the thought of the prerequisites of cloud envi-ronment for TRE, CoRE consolidates a few watchful plans and advancements to diminish CoRE's activity cost and enhance its TRE proficiency. In the first place, CoRE utilizes an impermanent little piece reserve for every customer to diminish the capacity cost of In-Chunk Max-Match at a server. It requires reserve syn-chronization between the sender and receiver, however it just identifies the traffic redundancy in here and now and in this manner cloud flexibility does not have much impact on the TRE effectiveness. Second, a solitary pass filtering calculation is utilized as a part of CoRE to decide lump limits in the TCP stream while in the meantime acquiring the fingerprints inside pieces used to discover maximal copy substrings in lumps. It proficiently coordinates two layers of TRE activities. Third, existing pre-expression based plan in PACK requires that the anticipated lump precisely shows up at the normal position of TCP stream. Indeed, even a little offset of the active information at the sender, contrasted and the information reserved at the receiver, can discredit the forecasts and incredibly debase TRE effi-ciency. To guarantee forecast proficiency against information transforms, we propose an enhanced expectation based TRE. In CoRE, the sender isolates the active information into lumps similarly as the receiver, and contrasts the marks of active pieces and all lump forecasts as of late got from the receiver paying little heed to their normal posi-tions in the TCP stream. At the receiver, a versatile predic-tion calculation is proposed, which progressively chooses the forecast window estimate, i.e., the quantity of resulting pieces to anticipate based on the hit proportion of chronicled lump expectations. Along these lines, CoRE increases preferable TRE productivity over PACK.

The execution of CoRE is assessed with a few traf-fic hints of this present reality applications. Our follow driven assessment comes about demonstrate that the dispersion of traffic redundancy at long haul and here and now time scales changes with various applications, which additionally causes distinctive CPU costs for CoRE at the cloud servers. A few applications may have prevailing long haul redundancy in traffic, while some have overwhelming here and now redundancy. In CoRE, recognizing here and now traffic redundancy has higher compu-tation cost than identifying long haul traffic redundancy. Along these lines, so as to enhance the advantage cost proportion between bandwidth reserve funds and TRE task costs, we additionally propose a versatile answer for find the overwhelming redundancy in the traffic and brilliantly choose which layer of TRE in CoRE to be empowered.

In outline, the commitments of this paper are recorded as takes after:

1. By a genuine follow driven examination, we distinguish the limita-tions of existing end-to-end TRE answers for captur-ing here and now and long haul redundancy of information traffic.

2. We propose a two-layer TRE conspire, named CoRE, to viably recognize both long haul and here and now traffic redundancy.

3. Several enhancements are proposed in CoRE to diminish the working cost, enhance the expectation proficiency and increment the advantage cost proportion of TRE, individually.

4. We execute CoRE and measure its advantages and costs based on broad analyses by utilizing sev-eral certifiable system traffic follows.

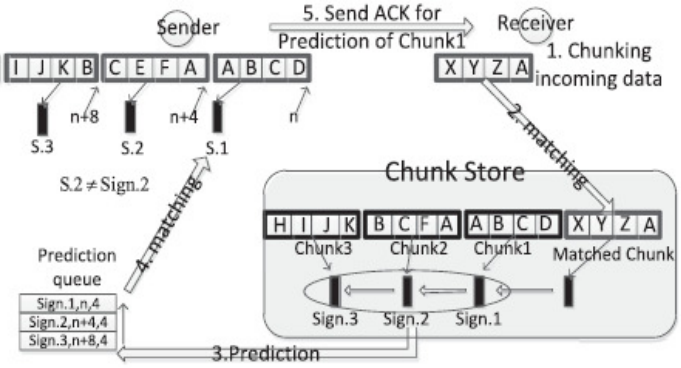

Fig. 1. A brief description for PACK algorithm. s:# and sign:# are chunk signatures. n þ # is the expected offset of a predicted chunk in TCP stream.

2. LITERATURE REVIEW

Eyal Zohar, Israel Cidon, and Osnat Mokryn [1]

Pack (Predictive ACKs), a traffic redundancy elimination solution is presented. A receiver based, end to end traffic redundancy elimination mechanism is provided to optimize bandwidth in cloud. Prediction at the receiver's end is made to eliminate traffic between the cloud and end to end users. A monitoring of the input stream is conducted then matching of the chunk is performed. Result of this long time chunk matching is sent to the sender's end. That content helps to make the decision about TRE operation. A hint about the TRE is sent to the sender. Sender perform TRE operation when that hint matched found. After the whole operation a prediction ACK is send to the receiver's end. At receiver's end a lightweight chunking scheme is used, which enhances the performance of computation speed and provide better performance to eliminate redundancy in the traffic.

Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P. C. Lee, Wenjing Lou [2]
Data deduplication is a technique which used to eliminate redundant data from the database and reduce storage space and bandwidth. An authorized deduplicate scheme is presented. That technique used an authorized redundancy elimination technique to optimize the bandwidth in cloud. In convergent encryption scheme user's original copy of the data is used generate convergent key and that convergent key used to encrypt the original copy. User also derive a tag for data copy, such that if two data copies are same then the tags of these copies are also same. That tag further used to remove duplicate copies of the data. That technique used to provide an authorized deduplication mechanism which used a token based scheme to eliminate duplicate data in a hybrid cloud scenario. That a secure outsourcing mechanism is provided to the user which enhances the functionality of the whole deduplication mechanism[7].

Lei Yu, Haiying Shen, Karan Sapra, Lin Ye and Zhipeng Cai [3]
Cloud computing provides various on-demand services to the user. In that various user access these services simultaneously that generate congestion in data transmission channel. Traffic redundancy elimination can be used to enhance the bandwidth capacity and also reduce cost of the bandwidth. But to eliminate TRE there either a sender based TRE

International Journal of Modern Engineering & Management Research | Vol 5 | Issue 1 | March 2017 6
Traffic Redundancy Elimination and Bandwidth Optimization Over Cloud Author(s): Iti Jain, Hitesh Gupta, Sweta Gupta, Dr. Vineet Richhariya elimination technique or receiver based TRE technique can be used. But these techniques are not efficient to provide better solution for TRE. CORE (Cooperative end to end traffic redundancy elimination) is presented, that technique uses both a sender and receiver based traffic redundancy elimination technique to provide an enhanced mechanism for TRE. In that technique cooperative operations between the sender and receiver is used to remove the redundant data from the bandwidth.

Swathi Kurunji, Tingjian Ge, Benyuan Liu, Cindy X. Chen [4]
A read optimized databases provides better performance for various read intensive data warehouse applications. In these applications data increases rapidly a flexible and dynamically environment like cloud is required to provide better performance to the user. But there is an efficient query mechanism is required to provide better performance to store data in these databases. In cloud scenario nodes are increases rapidly thus that generate inter-node communication cost that also generate a huge amount of data which requires an enhanced functionality to handle that data. A storage structure called PK-map and query processing technique is used to reduce the inter node communication cost. In that structure primary key and foreign key of the tables can be used to reduce the inter node communication from the various databases.

Lluis Pamies-Juarez, Pedro Garcia-Lopez, Marc Sanchez-Artigas, Blas Herrera [5]
"Towards the Design of Optimal Data Redundancy Schemes for Heterogeneous Cloud Storage Infrastructures" analyze how distributed redundancy schemes can be optimally deployed over heterogeneous infrastructures. Specifically, they are interested in infrastructures where nodes present different online availabilities. Considering these heterogeneities, they present a mechanism to measure data availability more precisely than existing works. Using this mechanism, they infer the optimal data placement policy that reduces the redundancy used, and then its associated overheads up to 70%.

A. Gupta, A. Akella, S. Seshan, S. Shenker, and J. Wang, [6]
Cloud computing brings significant benefits for service suppliers and users due to its characteristics: e.g., on demand, gets use, scalable computing. Virtualization management may be an important task to accomplish effective sharing of physical resources and scalability. Transmission price plays a crucial role once attempting to minimize cloud price. But for server specific TRE approach it's troublesome to

handle the traffic with efficiency and it doesn't suites for the cloud atmosphere due to high process prices. During this paper we have a tendency to provide a survey on the new traffic redundancy technique called novel-TRE conjointly called receiver based mostly TRE. This novel-TRE has vital options like police investigation the redundancy at the shopper, repeats seem enchained, matches incoming chunks with a antecedently received chunk chain or native file and causing to the server for predicting the longer term information and no want of server to ceaselessly maintain shopper state, our implementation maintains chains by keeping for Associate in Nursing chunk solely the last discovered ensuant chunk in an LRU fashion. So on the receiver aspect we are able to refresh the chunk store for incoming chunks.

Zhifeng Xiao and Yang Xiao,IEEE June 2013 conference – —Security and Privacy in Cloud Computing‖ [7]
They have worked on various attribute confidentiality, integrity, availability, accountability, and privacy-preservability and performed the various security concern issues in aspects, authors have systematically studied the security and privacy issues in cloud computing based on an attribute-driven methodology, We have identified the most representative security/privacy attributes (e.g., confidentiality, integrity, availability, accountability and privacy- preservability), as

International Journal of Modern Engineering & Management Research | Vol 5 | Issue 1 | March 2017 7
Traffic Redundancy Elimination and Bandwidth Optimization Over Cloud Author(s): Iti Jain, Hitesh Gupta, Sweta Gupta, Dr. Vineet Richhariya
well as discussing the vulnerabilities, which may be exploited by adversaries in order to perform various attacks. Defense strategies and suggestions were discussed as well, thus this is the paper included the security and study aspects in cloud computing, the data integrity verification made dealing with encryption algorithm and the audit was performed with the help of hashing algorithm available in order to verify the value generated again while checking the data integrity available with the associated file, here they have worked on different aspects such as user account access approach, availability of data, data changing or integrity verification and the technique should be privacy preserving so that the data should not be leak during the cloud execution.

3 RELATED WORK

Since significant redundancy has been found in the network traffic [8], [11], due to repeated accesses to the same or simi-lar data objects from the Internet end-users, several TRE techniques have been proposed to suppress duplicate data from the network transfers to reduce the bandwidth usage. A protocol-independent packet-level TRE solution was first proposed in [8]. In this work, the sender/receiver maintains a local cache respectively which stores recently transferred/ received packets. The sender computes the Rabin finger-prints [12] for each packet by applying a hash function to each 64 byte sub-string of the packet content, and selects a subset of representative fingerprints as to the packet con-tent. For an outgoing packet, the sender checks whether its representative fingerprints have appeared in earlier cached packets. If yes, the sender identifies the maximal duplicate region around each matched fingerprint and replaces the region with a fixed-size pointer into the cache to compress the packet. To decode compressed data, the receiver repla-ces the pointer with the corresponding referred data in its local cache. Several commercial vendors have developed such protocol-independent TRE algorithms into their "WAN optimization" middle-boxes [13], [14], [15] placed at either end of a WAN link. The successful deployment of TRE solutions in enterprise networks motivated the explo-ration of TRE deployment at routers across the entire Inter-net [16], [17]. In [16], redundancy-aware intra- and inter-domain routing algorithms are proposed to further enhance network-wide TRE benefits. In [17], an architecture for net-work-wide TRE is proposed, which allocates encoding and decoding operations across network elements and perform redundancy elimination in a coordinated manner.

The previous work on traffic redundancy [11] found that over 75 percent of redundancy was from intra-host traffic, which implies that an end-to-end solution is very feasible for redundancy elimination. Accordingly, a sender-based end-to-end TRE [9], named EndRE, was proposed for the enterprise networks. By maintaining a fully synchronized cache with each client at the server, EndRE offloads most processing effort and memory cost to servers and leaves the clients only simple pointer lookup operations. It suppresses duplicate byte strings at the order of 32-64B.

A receiver-based end-to-end TRE, PACK, is proposed for cloud environment [7], [10]. At the receiver, the incoming TCP data stream is divided into chunks. The chunks are linked in sequence, which forms a chain, and stored into a local chunk store. The receiver compares each incoming chunk to the chunk store. Once finding a matching chunk on a chain, it retrieves a number of subsequent chunks along the chain as the predicted chunks in the future incoming data. The signatures of the retrieved chunks and their expected offsets in the incoming data stream are sent in a PRED mes-sage to the sender as a prediction for the sender's subsequent outgoing data. Fig. 1 briefly describes the PACK algorithm. Once finding a match with an incoming chunk ½XYZA& in the chunk store, the receiver sends to the sender the triples of sig-nature, expected offset and length of following chunks ½ABCD&, ½BCFA& and ½HIJK& in the same chain, i.e., ðSign:1; n; 4Þ, ðSign:2; n þ 4; 4Þ and ðSign:3; n þ 8; 4Þ as pre-dictions. To match data with a prediction, the sender com-putes SHA-1 over the outgoing data at the expected offset with the length given by the prediction, and compares the result with the signature in the prediction. Upon a signa-ture match, the sender sends a PRED-ACK message to the receiver to tell it to copy the matched data from its local stor-age. In this way, PACK offloads the computational effort of TRE from the cloud servers to the clients, and thus avoids the additional computational and storage costs incurred by TRE at the cloud to outweigh the bandwidth saving gains.

The efficiency of PACK for capturing the long-term redundancy is susceptible to data changes, which can hap-pen frequently in various cloud applications involving fre-quent data update such as collaborative development [18] and data storage [19]. In PACK, the sender determines the data chunk to match based on the position and the length both given by the prediction. The prediction is true only if the predicted chunk exactly appears at the expected posi-tion in the byte stream. Thus, even a small position offset in the sender's outgoing data due to data insertion or dele-tion can invalidate all the following predictions. For exam-ple, in Fig. 1, 'E' is inserted into the sender's data. For the outgoing data ½ABCDBCEFAHIJK&, the PACK sender matches Sign:1 with the data chunk at the expected offset n with given length 4, which is ½ABCD&. Similarly, it will match Sign:2 with the data chunk at the expected offset n þ 4 with length 4 that is ½BCEF &, which causes a matching failure. Therefore, in PACK once a matching failure occurs, the following predictions are abandoned, even though the data chunk ½HIJK& can match with the third prediction. In CoRE, an improved prediction-based TRE is proposed. The CoRE sender performs the content-based chunking as the receiver, and finds a match by computing the signatures of the chunks and looking up them in a prediction store that keeps recently received predictions. In this way, CoRE achieves resiliency against data changes

TABLE 1
Characteristics of Internet Traffic Traces

Trace Name	Description	Dates/Start Times	Duration	Total Volume(MB)	IP Pairs	Servers(IP)	Clients(IP)
Univ-HTTP	Inbound/outbound http	10am on 11/05/11	120s	1900	8277	3183	1931
Univ-SN	Inbound/outbound for SN	3pm on 11/08/11	2h	1300	3237	6	894

simply deploys two separate TRE schemes together and switches from the receiver-based TRE to the sender-based TRE if data changes make predictions inefficient. Since at any time only one scheme works, the hybrid solution cannot cap-ture the long-term and short-term redundancy simulta-neously. It also cannot adaptively and flexibly distribute TRE effort among two separate schemes according to the charac-teristics of traffic redundancy. In contrast, with joint work of two layers of TRE, CoRE is able to simultaneously and adap-tively capture the long-term and short-term redundancy.

Besides the applications in wired networks, TRE also has been explored in wireless and cellular networks [20]. REfac-tor [20] aims to remove the traffic redundancy at the sub-packet level by IP-layer RE with content overhearing. It leverages overheard data by estimating the overhearing probability of data for the receiver and removing the chunks that highly likely have been overheard. Celleration [21] is a gateway-to-mobile TRE system for the data-intensive cellu-lar networks. The gateway predicts the future chunks to a mobile device and eliminates the transmission of the chunks whose predictions are confirmed by the mobile devices, which indicates the chunks are cached on the device. The measurements in [22] shows that TCP-level TRE saves sig-nificant bandwidth consumption for 3G traffic. In [23], Zhang et al. surveyed the existing protocol-independent TRE techniques and systems developed for wired and wire-less networks respectively. In this paper, we focus on the TRE for wired networks in the context of the cloud computing

4 Proposal work

CORE DESIGN

4.1 Overview

CoRE has two TRE modules each to capture here and now redundancy and long haul redundancy separately. Two TRE modules are coordinated into a two-layer redundancy recognition framework. For any outbound traffic from the server, CoRE initially identifies the long haul redundancy by the main layer TRE module. On the off chance that no redundancy is discovered, it swings to the second-layer TRE module to look for here and now redun-dancy at better granularity.

The primary layer TRE module identifies long haul redun-dancy by a forecast based Chunk-Match approach like PACK [7]. Thinking about the wastefulness of PACK portrayed in Section 2, we propose an enhanced expectation calculation to proficiently deal with information changes so a little information change won't influence the TRE productivity. In particular, the sender in CoRE plays out an indistinguishable lumping calculation from the receiver to isolate the active information into pieces. At that point, for each lump, the sender processes its mark (e.g., its SHA-1 hash esteem) and looks into the mark in the predic-tion store that keeps all forecasts as of late got from the receiver. On the off chance that a coordinating mark is discovered, the sender sends an expectation affirmation PRED-ACK message to the receiver rather than the active lump, regardless of whether the piece has the normal offset in TCP stream which is indicated in the forecast. This is rather than PACK. Along these lines, our expectation coordinating does not include information position in TCP stream, which rolls out CoRE versatile against information improvements.

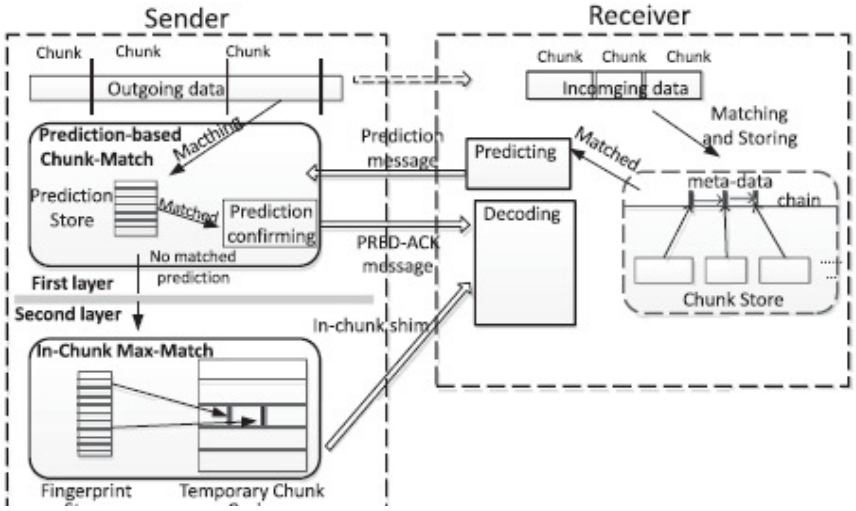

Fig:overview of core

In the second-layer TRE module, both the server and cli-ent maintain a temporary small local chunk cache to store most recently transmitted and received chunks during their communication respectively. The server matches outgoing data with the local cache at fine granularity to detect short-term redundancy. In particular, the sender stores its recently transferred chunks in its local chunk cache. For every chunk in the cache, the sender computes a set of representa-tive fingerprints, each of which is the hash value of a data window of size w in the chunk. Every representative finger-print (along with a pointer to the corresponding chunk in the cache) is stored into a fingerprint store. To detect the redundancy inside an outgoing chunk, the sender performs In-Chunk Max-Match to identify maximal sub-strings in the chunk which

have duplicates in the chunk cache. Specifi-cally, the sender compares each representative fingerprint of the outgoing chunk against the fingerprint store to check whether a matching fingerprint exists. A matching finger-print indicates that the outgoing chunk has a data window of size w which also appears in a previously transmitted chunk in the cache. If a matching fingerprint is found in the fingerprint store, the in-cache chunk which it points to is retrieved. The data window corresponding to the matching fingerprint in the in-cache chunk is expanded byte-by-byte in both directions and compared with the outgoing chunk, in order to identify the maximal overlap substring between the in-cache chunk and the outgoing chunk. Then, the sender encodes the maximal matched substring in the out-going chunk with an in-chunk shim which contains the signa-ture of the corresponding in-cache chunk, the offset and length of the matched substring.

For any incoming packet, the receiver first decodes in-chunk shims if any. To decode an in-chunk shim, the receiver retrieves the chunk in its local cache which has the same signature as that in the shim, and replace the shim by the substring of the in-cache chunk according to the offset and length specified by the shim. If the packet is a PRED-ACK message, the receiver checks the confirmed prediction in its prediction store and retrieves the corresponding pre-dicted chunk in its chunk store. Then, it copies the chunk to its TCP input buffer according to its offset in TCP stream specified by the PRED-ACK message. An overview of CoRE is shown in Fig. 4. Next, we explain the details of CoRE.

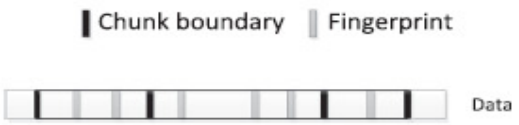

Fig. 5. Chunking and fingerprinting.

4.2 Chunking and Fingerprinting

As described above, CoRE coordinately uses prediction based Chunk-Match and In-Chunk Max-Match to implement two-layer TRE in order to maximally detect redundancy by captur-ing both long-term and short-term redundancy. Chunk-Match identifies the repeated chunks in a TCP stream while Max-Match identifies the maximal repeated substrings in a chunk. Similar to PACK [7], CoRE uses a large chunk size, i.e., at the order of several KB, for Chunk-Match. A larger chunk size can reduce the total number of chunks in a TCP stream, which further results in fewer expensive SHA-1 operations for computing signatures of chunks, less storage cost for stor-ing the meta-data of the chunk store at the receiver and also fewer prediction transmissions from the receiver. With a large chunk size, prediction based Chunk-Match identifies long-term redundancy with low operating cost. With computing the representative fingerprints at average interval of 32-64 bytes for any chunk, which is referred to as fingerprinting, In-Chunk Max-Match is able to identify redundancy inside chunks at a small granularity. In this way, when the transmis-sion of an outgoing chunk cannot be eliminated by prediction based Chunk-Match, In-Chunk Max-Match can capture the redundancy inside the chunk to improve bandwidth savings. As we can see, chucking and fingerprinting are basic opera-tions in CoRE and their efficiency is desired for CoRE to achieve low computation cost.

Chunking algorithms in previous works [7], [9], [11] determine chunk boundaries based on either byte or finger-print. Byte based algorithms, such as MAXP [11] and Sam-plebyte [9], choose a byte that satisfies a given condition as chunk boundary. The fingerprint-based algorithms, such as Rabin fingerprint based [9] and PACK chunking [7], com-pute fingerprints by applying a pseudo-random hash func-tion to sliding windows of w contiguous bytes in a data stream and select a subset of fingerprints with a given sam-pling frequency. The first byte in the window of each chosen fingerprint forms the boundaries of the chunks. As opposed to previous works [7], [8], [9], [11], [24] which use Chunk-Match and Max-Match exclusively, CoRE needs the sender to perform both chunking and fingerprinting to every chunk. A straight approach for this is to first use a chunking algo-rithm to divide a data stream into chunks and then computes the fingerprints for each chunk. However, this approach needs scanning the byte string twice for chunking and finger-printing respectively, which increases the computation cost of the servers. To save the cost, we can utilize the chunking algorithm based on the fingerprint to generate chunks and fingerprints of chunks within one single-pass scanning. Spe-cifically, based on XOR-based hash in the chunking algo-rithm of PACK [7], our algorithm computes fingerprints over the byte stream and chooses a subset of them as chunk boundaries with the remaining fingerprints as the finger-prints of chunks, as shown in Fig. 5.

In our algorithm, a XOR-based rolling hash function is used to compute a 64-bit pseudo-random hash value over each sliding window of w bytes of the byte stream. As shown in [7], the XOR-based rolling hash function achieves higher speed than Rabin fingerprinting algorithm for com-puting fingerprints. Therefore, we choose the XOR-based rolling hash function to generate fingerprints and chunk boundaries. Given k bit-positions in a 64 bit string, denoted by Pf ¼ fb1; b2; :::; bkg, if the 64-bit pseudo-random hash has '1' value at all these positions, it is chosen as a fingerprint. For each fingerprint, given a set of n bit-positions Pc in a 64-bit string such that jPcj ¼ n and Pf Pc, if the fingerprint has '1' value at all positions in Pc, it is chosen as a chunk boundary. As a result, the average sampling interval for fin-gerprints is 2k bytes and the average chunk size is 2n bytes. Algorithm 1 shows the pseudo-code of our chunking and fingerprinting algorithm with w ¼ 48; n ¼ 13; k ¼ 6, which gives the average fingerprint sampling interval of 64 B and the average chunk size of 8KB. Note that in CoRE only the sender needs to run both chunking and fingerprinting over the outgoing data stream while the receiver just runs chunk-ing over the received data.

Algorithm 1. Chunking and Fingerprinting

1: cmask 0x00008A3110583080; //13 1-bits, 8 KB chunks

2: fmask 0x0000000000383080; //6 1-bits, 64 B fingerprint sampling interval

3: longval 0; //64-bit

4: for all byte 2 stream do

5: shift left longval by 1 bit;

6: longval longval XOR byte;

7: if processed at least 48 bytes and

ðlongval AND fmaskÞ ¼¼ fmask then

8: found a fingerprint f;

9: if ðlongval AND cmaskÞ ¼¼ cmask then

10: f is a chunk boundary;

11: end if

12: end if

13: end for

4.3 CoRE Sender Algorithm

The sender uses a prediction store to cache the most recent predictions received from the receiver. Each prediction con-tains the SHA-1 signature of a predicted chunk and its expected offset, i.e., TCP sequence number in the TCP byte stream. The sender also has a chunk cache to store chunks that have been recently sent. A fingerprint store holds the meta-data of every representative fingerprint for each cached chunk, which includes the fingerprint value, the address of the chunk in the cache referred by the finger-print, and the byte offset of the window in the chunk over which the fingerprint is computed.

When receiving new data from the upper layer applica-tion, the sender performs the chunking and fingerprinting algorithm. For each outgoing chunk, if the prediction store is not empty, the sender first computes the SHA-1 signature of the chunk and then looks up it in the prediction store. If a matching signature is found in a prediction p, the sender replaces the outgoing chunk with a PRED-ACK confirma-tion message which carries a tuple < offsetp; offsets > where offsetp is the expected offset in prediction p and offsets is the actual offset of the outgoing chunk in the TCP stream. As we explain later in Section 4.4, each prediction is uniquely identified at the receiver by its expected offset.

In PACK, the sender simply discards previously received predictions which predict chunks before the position of cur-rent outgoing data in TCP stream. The chunk signature of a prediction is compared with the outgoing data chunk which is in the TCP sequence interval uniquely determined by the offset and length in the prediction. However, as we pointed earlier in Section 2, PACK suffers degraded TRE efficiency if the data offset changes due to disperse insertions and dele-tions of data. With consideration of possible data changes, the predicted chunks may actually appear in the near future after the expected offset. Thus, we improve the prediction-based algorithm by allowing the sender to use the historical predictions regardless of their expected chunk offset in TCP stream. To this end, the CoRE sender needs to divide the out-going data into chunks in the same way as the receiver does with a content-based chunking algorithm, instead of dividing according to the chunk offset and length specified by the receiver in the prediction. Each outgoing chunk is then com-pared with all entries in the prediction store regardless of their expected chunk positions (i.e., TCP sequences), such that the chunk can match a prediction having an inconsistent TCP sequence but the same signature. As a result, CoRE can achieve resiliency against data changes and be able to lever-age useful predictions from the receiver as much as possible. As for the example in Fig. 1, because the content around chunk boundaries does not change, CoRE divides the outgo-ing data at the sender into chunks ½ABCD& ½BCEFA& ½HIJK&, given the same chunking algorithm as at the receiver that determines chunk boundaries based on data content. The sig-nature of chunk ½HIJK& is already received as a prediction from the receiver. Then, CoRE would find a match and com-press the chunk. By contrast, PACK misses this opportunity as mentioned before in Section 2.

If the prediction store is empty or the prediction-based Chunk-Match does not find a matching prediction, the sender then performs In-Chunk Max-Match. It checks the fingerprints of the outgoing chunk against the fingerprint store. If a matching fingerprint is found, the corresponding chunk is retrieved from the chunk cache. Considering the possible collision in the fingerprint namespace due to the same hash values for different data, the window corre-sponding to the matching fingerprint in the chunk is com-pared with the outgoing chunk byte-by-byte. Then, if the window has the same data, it is expanded byte-by-byte in both directions to obtain the maximal overlapped substring. Each matching substring in the outgoing chunk is encoded with a shim < signc, length, offsetc, offsets > where signc is the signature of the in-cache chunk, length is the length of the matching substring, offsetc is the offset of the matching substring in the in-cache chunk, and offsets is the offset of the matching substring in the outgoing TCP stream. If an in-cache chunk has multiple matching substrings with an out-going chunk, all corresponding shims can be compressed together in the form of < signc, length1, offsetc1, offsets1, length2, offsetc2, offsets2, . . . > where each matching sub-string i is described by lengthi, offsetci and offsetsi.

After the sender sends out the chunk, the sender updates the chunk cache and the fingerprint store with this chunk. Fig. 6 describes the sender algorithm for processing outgoing data by state machines. The details for the maintenance of local data structures at the sender are described as follows.

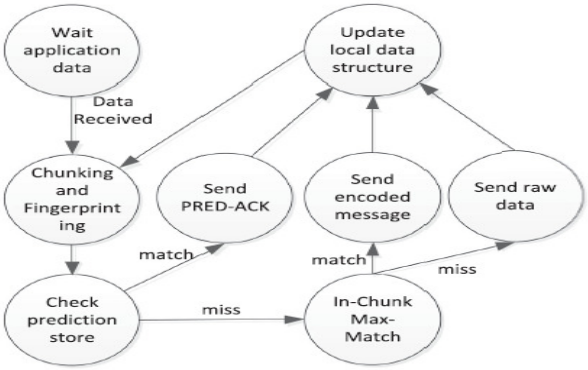

Fig. 6. Sender algorithm.

Chunk Cache. The chunk cache is a fixed-size circular FIFO buffer. The server maintains a chunk cache for every client. If no further request is received from the client within a period, the cache is released. Once receiving the request from a new client, the server allocates a chunk cache for the client. The chunk cache stores recently transferred chunks from the server to the client, via either one TCP connection or multiple TCP connections. Once the chunk cache is full, the earliest chunk is evicted and the fingerprints pointing to it are invalidated. Each entry in the chunk cache is a tuple< data; signature > , where data field is the chunk data and signature field is its SHA-1 hash. Note that the signa-ture is not always computed for all chunks. The signature field is filled for a chunk in two cases. First, the outgoing chunk's signature has to be computed for prediction match-ing when the prediction store is not empty, so in this case the chunk is stored into the cache with its signature. Second, during In-Chunk Max-Match, if the in-cache chunk where the maximal matching substring is obtained has empty the signature field, its signature is computed and filled. In this way, the expensive signature computation is only per-formed on demand, which avoids unnecessary SHA-1 com-putations to reduce the server's computation cost.

Since the chunk cache is used to detect short-term traffic redundancy, a small cache size can be sufficient. As indi-cated in Section 3, even with small cache size like 250 KB, significant short-term redundancy is detected by Max-Match of the sender-based TRE. In CoRE, the server and the client also need to maintain synchronized chunk caches for In-Chunk Max-Match. The changes of service point in the cloud may cause the loss of cache synchronization at the server. However, since the small chunk cache only stores short-term historical traffic, the changes of service point just result in the missing of traffic redundancy in a short term, and its impact on the TRE efficiency is limited. Besides using small chunk cache, CoRE releases the cache if no fur-ther service request is received from the client within a period, which further reduces the storage cost of CoRE at the server.

Prediction Store. As opposed to that a chunk cache is shared by all TCP connections to the same client which the cache is allocated for, a prediction store is allocated by the server for each TCP connection with a client. The expected chunk offset of a prediction is represented by the TCP sequence number of the predicted chunk (i.e., the TCP sequence number of the first byte in the chunk) in the TCP stream. Once receiving a new prediction from the receiver, the sender inserts it into its prediction store associ-ated with the corresponding TCP connection. The predic-tion store holds the most recent predictions. Outdated predictions need to be identified and evicted to limit the size of the prediction store and ensure the efficiency of pre-diction matching. In CoRE we define the elapsed time of a prediction p in the store as:

$$E_p = \begin{cases} 0, & Seq_c \leq_{seq} Seq_p \\ (Seq_c - Seq_p) \bmod 2^{32}, & Seq_c >_{seq} Seq_p \end{cases} \quad (1)$$

where Seqc is the TCP sequence number of the chunk cur-rently to be sent, and Seqp is the expected TCP sequence number in the prediction p. seq and > seq are comparisons of TCP sequence number with modulo 232 [25]. We use TTLpred to denote the maximum elapsed time of a prediction, a sys-tem parameter. At the time when CoRE compares the outgo-ing chunk having the TCP sequence number Seqc against the prediction store, the predictions satisfying Ep > TTLpred are removed from the prediction store. In this way, we use the difference of TCP sequence number between a prediction and current outgoing data to measure the timeliness of the prediction, given that a larger difference implies the less rele-vance of the prediction with the outgoing data. By keeping all the predictions within TTLpred rather than only the latest prediction as in PACK, CoRE has higher chance to find the matches since the chunks predicted by these predictions may appear in the near future due to their delayed appear-ance or repetitive appearance in the TCP stream.

4.4 CoRE Receiver Algorithm

For each TCP connection, the receiver divides the incoming data stream into chunks and maintains a local prediction store which caches recent predictions for the TCP connec-tion. To reconstruct the raw data stream, the receiver pro-cesses incoming TCP segments according to their different types. There are three types of TCP segments from the sender to the receiver: PRED-ACK message, the message encoded with shims and raw data. Upon receiving a PRED-ACK message containing < offsetp; offsets > , the receiver first checks the corresponding prediction store to find the prediction p which has the expected offset offsetp. Then, the receiver retrieves the chunk predicted by prediction p from the chunk store and place it to the input buffer of the corre-sponding TCP connection with the offset offsets in the TCP stream specified by the sender. If receiving the message con-taining shim < signc, offsetc, length, offsets > , the receiver finds the chunk having signature signc from its chunk store. The matching substring in the chunk indicated by offsetc and length is copied to the receiver's TCP input buffer at the position offsets in TCP stream.

The chunking algorithm at the receiver is the same as Algorithm 1 without the fingerprint identification with

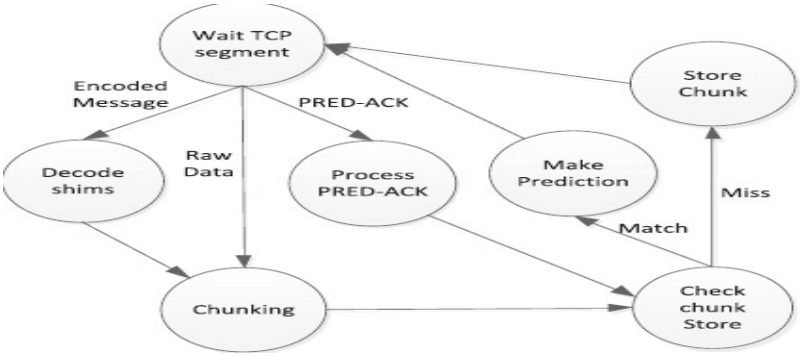

Fig. 7. Receiver algorithm.

fmask. The receiver maintains a local chunk store as in PACK [7], which stores chunks from all its TCP connections. The chunks from the same TCP connection are linked together in the order they are received and then stored into the chunk store. If the signature of an incoming chunk is found in the chunk store, the receiver makes one or multiple chunk predictions for a number of subsequent chunks from the sender. Each chunk prediction consists of the signature of the predicted chunk and its expected offset in the TCP stream. For any chunk prediction, if the predicted chunk has overlapped TCP sequence range with any previous pre-dictions in the corresponding prediction store, the predic-tion is discarded. Accordingly, within a TCP connection, the predictions from the receiver to the sender have no over-lap with each other on the TCP sequence range for their pre-dicted chunks. The chunk predictions are sent within a PRED message and in the order of their expected offsets. Fig. 7 shows the receiver algorithm, in which the receiver enters different processing procedures according to the types of incoming messages.

Since in a TCP connection the chunk predictions have no overlapped TCP sequence range for their predicted chunks, each prediction for the same TCP stream can be uniquely identified by the expected offset in it. Then, each entry in the prediction store consists of an expected TCP sequence number and a pointer to the corresponding predicted chunk. The receiver also periodically removes the outdated predictions from the prediction store in the similar way as the sender does. Each time when the receiver receives a chunk having the TCP sequence number Seqc, any predic-tion p with its elapsed time Ep (computed by (1)) greater than TTLpred is removed from the prediction store before the receiver tries to make any new predictions.

CONCLUSION

By the genuine follow driven investigation on existing end-to-end sender-side and receiver-side TRE arrangements, we distinguish their confinements for catching redundancy in here and now and long haul information redundancy. In this way, we propose a Coop-erative end-to-end TRE CoRE, which coordinates two-layer TRE endeavors and a few advancements for TRE effectiveness. We additionally propose a few improvement answers for CoRE by adaptively changing the forecast window measure and choose which layer of TRE is empowered by the distri-bution of traffic redundancy of the cloud application. Through broad follow driven examinations, we demonstrate that CoRE can productively catch both here and now and long haul redundancy, and can dispense with considerably more redundancy than PACK while bringing about a low extra task cost. Our assessment comes about demonstrate that distinctive disseminations of traffic redundancy at here and now and long haul time scales can cause diverse CPU costs for CoRE and versatile solu-tions for CoRE are fundamental for enhancing TRE productivity and lessening the task cost.

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, Apr. 2010.

[2] N. R. Herbst, S. Kounev, and R. Reussner, “Elasticity in cloud computing: What it is, and what it is not,” in Proc. ICAC. San Jose, CA: USENIX,2013, pp. 23–27.

[3] “Netflix,” [Online]. Available: www.netflix.com, (accessed on 17-Jun-2016).

[4] “Amazon elastic compute cloud (EC2),” [Online]. Available: http://aws.amazon.com, (accessed on 17-Jun-2016).

[5] “The hidden cost of the cloud: Bandwidth charges,” [Online].

Available: http://gigaom.com/2009/07/17/the-hidden-cost-of-the-cloud-bandwidth-charges, (accessed on 17-Jun-2016).

[6] “The real costs of cloud computing,” [Online]. Available: http:// www.computerworld.com/article/2550226/data-center/the-real-costs-of-cloud-computing.html, (accessed on 17-Jun-2016).

[7] E. Zohar, I. Cidon, and O. O. Mokryn, “The power of prediction: cloud bandwidth and cost reduction,” in Proc. ACM Spec. Int. Group Data Commun., 2011, pp. 86–97.

[8] N. T. Spring and D. Wetherall, "A protocol-independent tech-nique for eliminating redundant network traffic," in Proc. ACM Spec. Int. Group Data Commun., 2000, pp. 87–95.

[9] B. Agarwal, 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. National Spatial Data Infrastruct., 2010, pp. 419–432.

[10] E. Zohar, I. Cidon, and O. Mokryn, "Pack: Prediction-based cloud bandwidth and cost reduction system," IEEE/ACM Trans. Netw., vol. 22, no. 1, pp. 39–51, Feb. 2014.

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

[12] M. Rabin, "Fingerprinting by random polynomials," Technical report Harvard University, Cambridge, MIT, vol. TR-15-81, 1981.

[13] "Riverbed networks : Wan optimization." [Online]. Available: http://www.riverbed.com/solutions/wan-optimization.html, (accessed on 17-Jun-2016).

[14] "Juniper networks: Application acceleration." [Online]. Available: http://www.juniper.net/techpubs/en_US/release-independent/ jwos/information-products/pathway-pages/wx-series/product/ , (accessed on 17-Jun-2016)

[15] "Cisco wide area application services," [Online]. Available: http://www.cisco.com/c/en/us/products/routers/wide-area-application-services-waas-software/index.html, (accessed on 17-Jun-2016).

[16] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," in Proc. ACM Spec. Int. Group Data Commun. ACM, 2008, pp. 219–230.

[17] A. Anand, V. Sekar, and A. Akella, "Smartre: an architecture for coordinated network-wide redundancy elimination," in Proc. ACM Spec. Int. Group Data Commun. ACM, 2009, pp. 87–98.

[18] "Paas." [Online]. Available: http://www.ibm.com/cloud-com-puting/us/en/paas.html, (accessed on 17-Jun-2016).

[19] "Dropbox." [Online]. Available: www.dropbox.com, (accessed on 17-Jun-2016).

[20] S.-H. Shen, A. Gember, A. Anand, and A. Akella, "Refactor-ing content overhearing to improve wireless performance," in Proc. 17th Annu. Int. Conf. Mobile Comput. Netw., ser. MobiCom '11. New York, NY, USA: ACM, 2011, pp. 217–228. [Online]. Avail-able: http://doi.acm.org/10.1145/2030613.2030638

[21] E. Zohar, I. Cidon, and O. O. Mokryn, "Celleration: Loss-resilient traffic redundancy elimination for cellular data," in Proc. Twelfth Workshop Mobile Comput. Syst. & Appl., ser. HotMobile '12. New York, NY, USA: ACM, 2012, pp. 10:1–10:6. [Online]. Avail-able: http://doi.acm.org/10.1145/2162081.2162096

[22] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park, "Comparison of caching strategies in modern cellular backhaul networks," in Proc.11th Annu. Int. Conf. Mobile Syst., Appl. Serv., ser. MobiSys '13. New York, NY, USA: ACM, 2013, pp. 319–332. [Online]. Available: http://doi.acm.org/10.1145/ 2462456.2464442

[23] Y. Zhang and N. Ansari, "On protocol-independent data redun-dancy elimination," Commun. Surv. Tutorials, IEEE, vol. 16, no. 1, pp. 455–472, Feb. 2014.

[24] S. Ihm, K. Park, and V. S. Pai, "Wide-area network acceleration for the developing world," in Proc. USENIX annual technical confer-ence. USENIX Association, 2010, pp. 18–18.

[25] G. Wright and W. Stevens., TCP/IP Illustrated, Volume2: The Implementation., Boston, Massachusetts: Addison-Wesley, 1995.

[26] L. Yu, K. Sapra, H. Shen, and L. Ye, "Cooperative end-to-end traf-fic redundancy elimination for reducing cloud bandwidth cost," in Proc. of Int. Conf. Netw. Protocols, ser. ICNP '12. Washington, DC, USA:, 2012, pp. 1–10. [Online]. Available: http://dx.doi.org/ 10.1109/ICNP.2012.6459973

[27] "Pack source code,"[Online]. Available: http://www.venus-c.eu/ pages/partner.aspx?id=10, (accessed on 17-Jun-2016).

[28] "netfilter/iptables:libnetfilter_queue," [Online]. Available: http:// www.netfilter.org/projects/libnetfilter_queue, Oct 2005