A Marketplace for Efficient and Secure Caching for IoT Applications in 5G Networks

Muhammad Usman*, Muhammad Rizwan Asghar†, Imran Shafique Ansari‡, Fabrizio Granelli*,
Qammer H. Abbasi‡§, and Khalid Qaraqe‡
*Department of Information Engineering and Computer Science, University of Trento, 38123 Trento, Italy
†Cyber Security Foundry, The University of Auckland, 1142 Auckland, New Zealand
‡Department of Electrical and Computer Engineering, Texas A&M University at Qatar, Doha, Qatar
§School of Engineering, University of Glasgow. G12 8QQ Glasgow, U.K.
Email: {muhammad.usman, fabrizio.granelli}@unitn.it, {imran.ansari, khalid.qaraqe}@qatar.tamu.edu
r.asghar@auckland.ac.nz, qammer.abbasi@glasgow.ac.uk

***Abstract*—As the communication industry is progressing towards fifth generation (5G) of cellular networks, the traffic it carries is also shifting from high data rate traffic from cellular users to a mixture of high data rate and low data rate traffic from Internet of Things (IoT) applications. Moreover, the need to efficiently access Internet data is also increasing across 5G networks. Caching contents at the network edge is considered as a promising approach to reduce the delivery time. In this paper, we propose a marketplace for providing a number of caching options for a broad range of applications. In addition, we propose a security scheme to secure the caching contents with a simultaneous potential of reducing the duplicate contents from the caching server by dividing a file into smaller chunks. We model different caching scenarios in NS-3 and present the performance evaluation of our proposal in terms of latency and throughput gains for various chunk sizes.**

***Index Terms*—5G networks, IoT, Privacy, Security, Content caching, Duplicate contents, Convergent encryption**

I. Introduction

Fifth generation (5G) of cellular networks is taking shape in serving variety of applications with diverse Quality of Service (QoS) requirements. These applications range from high data rate video streaming to low data rate Internet of Things (IoT) communication [1]. Although IoT applications usually generate low data rate traffic, however, the accumulation of data from billions of devices can potentially put burden on the backhaul network. Concerning this, caching at the edge of a network can potentially solve this problem.

Caching at the network edge to reduce the delivery time of the content is already proposed in literature. Content Delivery Networks (CDNs) are one approach, which are widely adopted across the world to reduce the network latency in delivering web content. On the other hand, fog computing or edge computing is an approach that introduces resource-rich computation nodes near the end users. These nodes are mainly designed to provide fast computations at the edge of a network but can also be employed as caching servers, especially in wireless environments [2], [3]. Device-to-device (D2D) communication is another possible place for contents to be cached in wireless environments [4]. Other possibilities include Small cell Base Station (SBS) caching and Macro cell Base Station (MBS) caching [2].

In this paper, we discuss various caching possibilities in future 5G networks for a broad range of applications, including IoT and normal cellular users. To achieve this, we present a marketplace for 5G service providers and content providers, where 5G service providers can offer caching servers while content providers can use those caching servers to store the content. In addition to providing security of the cached contents, we also consider the possibility of reducing the Internet traffic by uniting the duplicate entries in the caching server. In this regard, we employ convergent encryption, a scheme that encrypts data with its own hash code, to manage the duplicate contents. Duplicate contents are difficult to manage by other end-to-end encryption schemes, such as blind cache [5], designed for caching environments. In those schemes, the encryption of the data being cached restricts the caching server to identify the multiple entries of the same content.

The main contributions of our work are listed as follows.

- We propose different caching possibilities for end users and IoT applications, apart from those mentioned in [2]. To achieve this, we present a marketplace for both 5G service providers and content providers for offering caching servers and caching contents, respectively.
- We employ convergent encryption as an end-to-end encryption scheme for aforementioned caching environments, which works to manage the duplicate contents as well, saving not only storage at the caching server but also the bandwidth by reducing the number of requests for the same content between caching server and the content provider.
- For handling data duplication, we propose to split the content into smaller chunks so that a part of the content can also be prevented from duplicate storage/requests.
- We measure the latency and throughput for all caching environments by varying the chunk size of the content and taking into account the encryption and decryption of each chunk.

The rest of the paper is organized as follows. Section II provides a problem statement to present a gap in the literature. Section III demonstrates the design overview and the key

idea presented in this work. Section IV reports performance analysis. Related work is reviewed in Section V followed by a discussion in Section VI. Finally, we draw some conclusions and highlight research directions for future work in Section VII.

II. Problem Statement

The motivation behind caching contents near end users is to reduce network latency, as many applications require fast processing and access to their stored data. This holds true for many IoT applications wherein billions of devices connected to the Internet are generating low rate data of measurements that many end users or applications request frequently. For instance, there are many scenarios wherein a large number of end users run applications that request similar IoT data, such as weather condition and monitoring, among many others. Caching these contents near end users or applications not only reduces the network latency but also reduces the load on the Internet by reducing the number of requests traversing through IoT cloud service providers.

On one hand, caching contents results in network bandwidth, throughput, and latency gains; on the other hand, caching contents on untrusted servers can raise serious privacy and security concerns. For instance, CDNs are abundantly utilized to improve the delivery of the contents by replicating them to the caching servers located geographically near to the end users. However, current CDN technology requires user contents and traffic to be exposed to CDN providers [5], thus compromising user's privacy and security. To address this, a secure solution for caching contents is required, which not only works for IoT applications but also for a range of applications and users using the Internet. End-to-end encryption, such as HTTPS, *i.e.,* Secure Sockets Layer (SSL) or Transport Layer Security (TLS) used with Hyper Text Transfer Protocol (HTTP), is thought as one of the solutions to secure the access of data across the Internet. However, when it comes to caching, the end-to-end encryption restricts the inline transparent caching by the network service provider to serve the previously requested content. Since SSL/TLS works between endpoints, *i.e.,* client and server, aiming to mitigate man-in-the-middle attack, it becomes challenging to secure data when it is stored in a middle server.

Out-of-band cache *a.k.a.* blind cache is one solution proposed by the industry [5] to overcome this problem. Blind cache is an encryption scheme, which shares the key of the encrypted data with the client and places encrypted data to the CDN server (see Fig. 1). This allows content providers to share the encryption key of the cached content directly with the client while content is accessible through the CDN network. The problem with this solution is the overlapping contents. As CDN is not exposed to the encrypted content so it can possibly save multiple copies of the same content accessed by different users. Moreover, the blind cache solution is proposed for CDNs, which is not only expensive in terms of budget but also requires a prior contract with a CDN provider.

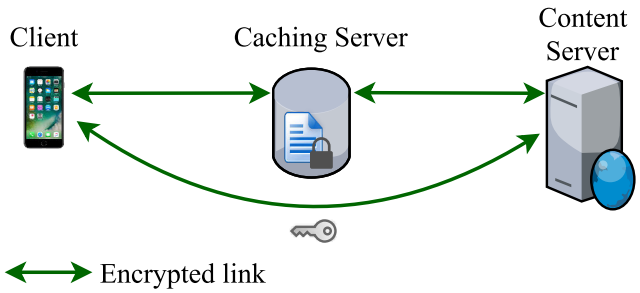

Fig. 1: Blind Cache: The key is shared to the client on a direct encrypted link between the content server and client. The encrypted data is shared with the caching server and a link to the caching server is sent to the client.

III. Design Overview

In this paper, we present a model wherein any intermediate 5G node, capable of storing the contents, can cache it. This, in turn, boosts the marketplace, where any node between client and server can act as a CDN. This section highlights the design overview of the proposed solution. Fig. 2 presents potential options to cache the contents near the end users. As we can see, we can cache at the user's premises, 5G cellular networks or CDNs. The entities presented in the Fig. 2 are defined as follows.

- **D2D Nodes:** D2D nodes are smart devices, such as smartphones, in the vicinity of end users that can be utilized to temporarily cache the contents.
- **Cloudlet:** Cloudlet represents the resource-rich computing resources in the vicinity of the end users [6]. It can be a Local Area Network (LAN) or even a single computer capable of caching the content for end users.
- **IoT Gateway:** An IoT gateway acts as a bridge between the IoT devices and the cloud to transport the information to and fro the cloud. It can be utilized as a potential location for caching contents.
- **5G Intermediate Nodes:** This represents any node in 5G networks, capable of storing data and providing it to the end users. The location of this node can be in the access network with the base station or in the core network. For experimental analysis, we consider this node in the access network.
- **CDN:** CDN is a well-known technology employed to reduce the network latency by caching the contents on the server located at the network edge, close to end-users.
- **Content Provider:** This represents the actual content provider. For instance, it can be an IoT cloud server providing information regarding vacant parking spaces around the city or a server streaming live videos to end users or a server providing weather information after collecting the same from various different IoT sensors.

Once a content provider receives a certain number of requests from the same region for the same content, it caches the content in one of the aforementioned servers, located as close

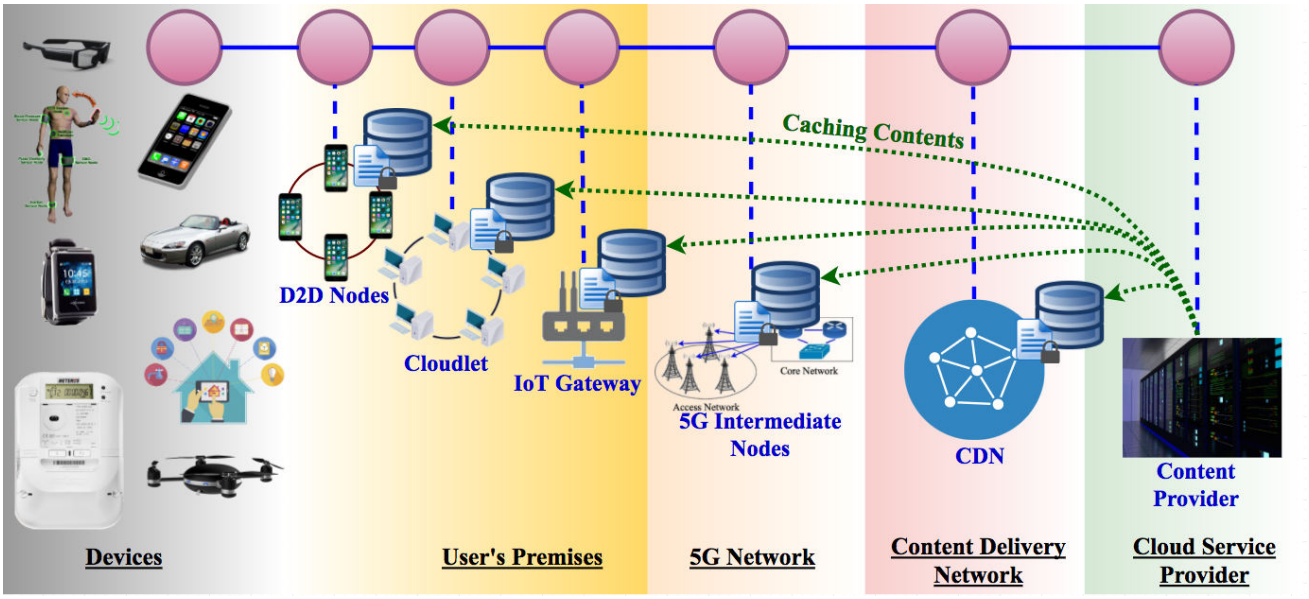

Fig. 2: Different caching options for a content provider: The data can be cached in any appropriate server between the content provider and clients. The choice of caching server depends upon the requirement of the application or end user, requesting the content.

to the end user as possible. To this end, the content can even be cached within the user's premises (see Fig. 2). Before caching the content, the content provider divides the content into smaller chunks. The motivation behind dividing the content into smaller chunks is the careful management of overlapping contents. There are many scenarios wherein some users request a content, while others request a part of it. For instance, some users in a region request for a complete book, while others in the same region request for specific chapters. If the content is already cached in the form of chunks, where each chunk corresponds to an individual chapter, then the request for separate chapters can easily be handled by the caching servers. The content provider just needs to communicate the unique identifier of the respective chunk (chapter in this case) to the client. Similar instances of overlapping contents include, but are not limited to, a movie season and its episodes, an album of images and separate images, and results from a search engine for similar queries, *etc.* Dividing a file into smaller chunks not only saves bandwidth in the Wide Area Network (WAN), but also saves storage at the caching servers.

The workflow of our solution is as follows. Upon receiving a request from a client for a particular content, the content provider establishes a secure channel with the client by creating a session key and checks if the content is already present in the caching server. If the content is not already cached but requested frequently, the content provider partitions the content into smaller chunks and calculates a hash code for each chunk. Then, each chunk is encrypted with its own hash code. Technically, we employ convergent encryption [7]. After this, the content is sent to the caching server, *e.g.,* a 5G intermediate node, to be accessed by the client. Then, the content provider sends a list to the client, containing all the hash codes, the encrypted chunk IDs, and the web address of the caching server. In case the content is already cached, the content provider simply shares the aforementioned list with the client through the secure channel protected via a session key.

Henceforth, in this paper, we evaluate the performance of each caching option in terms of latency and throughput. Moreover, we provide an efficient solution to secure the cached contents to be exposed to the caching server, which leads to preserving privacy of not only end users but also content providers. The partition of the content into smaller chunks solves the problem of caching multiple copies of overlapping content and convergent encryption makes our solution lightweight thereby making it best suitable for a wide range of scenarios, such as IoT applications and much more.

IV. Performance Analysis

In this section, we analyze performance of our proposed caching scheme in terms of latency and throughput gains. We used the NS-3 network simulator for simulating the proposed model presented in Section III. To cover the broad range of file sizes, we consider three distinct files types including text, image, and video. For simulations, we consider client as a smartphone requesting aforementioned data files from a content provider. We analyze all the caching options presented in Fig. 2 to evaluate latency and throughput gains, considering various chunk sizes for each file type. While estimating throughput, we also include the overhead of partitioning the file into smaller chunks inclusive of encrypting and decrypting each chunk of data.

Fig. 3 demonstrates the network latency from the client to each caching server. It can be noticed, the nearer the caching

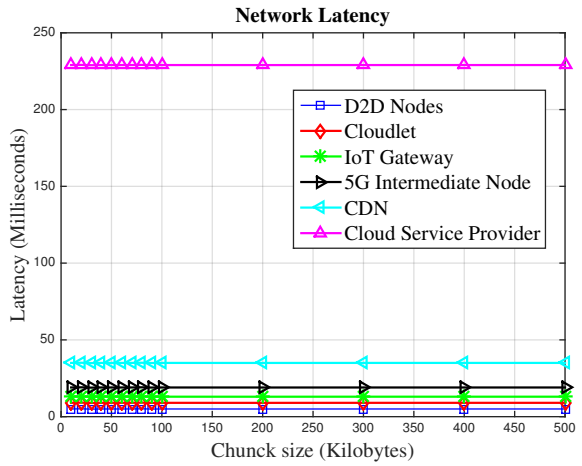

Fig. 3: Latency of various caching servers: The closer the server is to the client, the lesser is the latency.

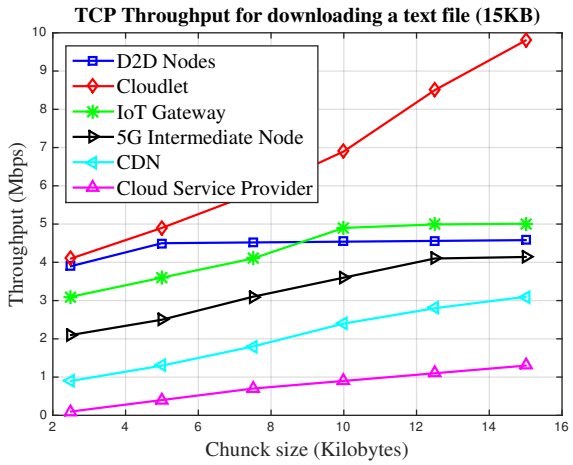

Fig. 4: Throughput of downloading a text file from different caching servers for varying chunk sizes: The throughput is always less than the optimal when the chunk size is less than the BDP of the link. Moreover, the throughput is, generally, directly proportional to the chunk size, *i.e.,* higher the chunk size, better is the throughput.

server, the lesser time it takes to serve the client. Moreover, the latency remains independent of the chunk size, which appears true as latency is the property of the network and the nodes involved. It has no relation to the size of the data being transferred between the server and the client. The D2D nodes provide the best latency. However, these resource-constrained caching servers (D2D nodes) cannot store much information nor can they store it for longer durations. Besides this, the mobility factor of D2D nodes makes them lesser suitable for adoption as caching servers. However, there are many scenarios wherein D2D nodes can be exploited as potential caching servers. For instance, remote health care applications, wherein the body sensors or implants do not generate huge amount of data, can exploit smartphone of the concerned doctor to cache the sensors' information of multiple patients a priori, instead of having to establish a one-to-one connection between the doctor and the patients and/or the cloud service provider that is responsible for storing IoT data from various patients.

Next, we analyze the throughput of the link between the client and each caching server for all aforementioned file types. With throughput, we refer to the TCP throughout between source and destination. For simulations, the standard size of a three-page text file is assumed as 15KB [8], the image is considered as 5MB [9], and the video file size as 50MB for a 10 minutes video of 360p resolution [10]. Fig. 4 represents the TCP throughput of accessing a text file from different caching servers. The throughput varies with the chunk size the file is actually divided into. It is evident from the Fig. 4 that the throughput is directly proportional to the chunk size. This is quite expected as increasing the chunk size decreases the number of chunks the file is actually divided into, which consequently decreases the combined overhead of encrypting and decrypting all chunks. Subsequently, an increase in the throughput is observed.

Moreover, it can be observed from Fig. 4 that caching

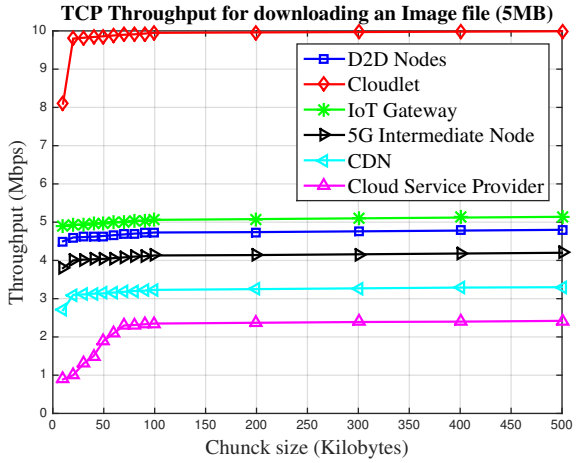

Fig. 5: Throughput of downloading an image file from different caching servers for varying chunk sizes: Here, too, the throughput is less than the optimal when the chunk size is less than BDP of the link.

servers located physically close to the end users provide higher values of the throughput. For example, the maximum throughput observed in the case of content providers is about 1.2 Mbps even if the file is not divided into any chunk. This is due to the reason that the network latency of a link to the content provider is around 230 milliseconds, which results in a higher Bandwidth Delay Product (BDP). For instance, in our scenario, the BDP of the content provider link turns out to be around 60 KB. It is important to note that the aforementioned estimated value of BDP is for the simulation scenario only. For real life scenarios, the BDP is quite inconsistent for a given flow and mainly depends on the network state, which

varies with network congestion, buffer overflow, and queuing *etc.*. Smaller chunk sizes, such as 5KB or 10 KB restricts the data pipeline to be fully occupied with data thereby reducing the throughput. On the other hand, low latency links, such D2D nodes, constitute a lesser value of BDP, which allows even smaller chunk sizes to fill the data pipeline completely. For instance, a latency of 6 milliseconds between a client and a D2D cache server with 5Mbps data rate gives a BDP of 3KB. Therefore, a chunk size of 2KB provides much lesser throughput as compared to a 5KB chunk. Subsequent chunk sizes do not provide much throughput improvement on D2D links.

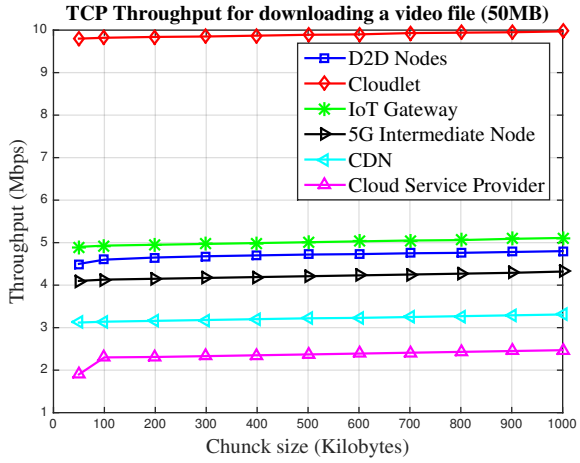

Fig. 6: Throughput of downloading a video file from different caching servers for varying chunk sizes: Here, too, the throughput is less than the optimal when the chunk size is less than BDP of the link.

Fig. 5 and Fig. 6 present the TCP throughput for scenarios wherein an image file and a video file, respectively, is downloaded from the caching servers, as discussed in Fig. 2. The throughput is, generally, directly proportional to the chunk size. However, there is an abrupt increase in the throughput when the chunk size surpasses the BDP of the flow.

V. Related Work

One of the most relevant works to our proposal is blind cache [5]. The authors in [5] propose an out-of-band caching scheme for HTTPS traffic. The encrypted data is cached by CDNs while the encryption key is directly provided to the client out-of-band. The idea of an out-of-band caching is very interesting, but there are certain concerns with this approach. The most significant limitation of this approach is the inefficient management of duplicate contents. As the content is end-to-end encrypted, CDNs will store multiple copies of it if requested by multiple users in the region of same CDN server. The second limitation is the delivery of encrypted contents to users using lossy or unstable networks, such as wireless networks with weak signal strength or partial coverage. To re-establish a lost connection, the content provider has to start a new secure session with a new session key. Moreover, new encryption keys are communicated with the client and the same data has to be cached again with new encryption keys. This is an inefficient utilization of communication and storage resources. On the other hand, our solution of dividing the content into smaller chunks and encrypting each chunk with its own hash code avoids the need of caching data again and again. Moreover, the link can be re-established from where it was interrupted (which depends on the chunk size) due to disconnection or any other similar reason.

The problem of reusability of cached contents is partially addressed by Leguay *et al.* in [11] by introducing a concept called *Cryptocache*. The authors propose to cache the contents based on pseudo-identifiers instead of using real identifiers. However, their solution does not solve the problem of data duplication. Moreover, their solution is not lightweight to be adopted by resource-constrained devices such as a smartphone with a lowly average computational power or IoT devices.

In [12], Mosko *et al.* provide a solution for secure caching in all-encrypted web. Their solution is analogous to blind caching solution proposed in [5]. This solution has same limitations of data duplication and reusability but is lightweight as compared to blind caching solution.

In [13], Yuan *et al.* propose an in-network caching scheme for delivering video files to end-users. The authors propose a request handler (dispatcher) in the network, which has the ability to identify, locate, and manage the in-network caching chunks. The main features of their solution include cache management and adaptive video delivery. However, the authors present their model for video files only and do not discuss the applicability of their approach to a mixed network traffic, more specifically, the case of IoT.

In [14], Engelmann and Elia exploit coded caching to preserve the privacy of end users requesting cached contents. The key features of their solution are confidentiality (user to content linking is not possible) and hiding the popularity statistics of the cached contents. However, the authors do not discuss the applicability of their approach to lossy or unstable networks wherein the connection can break very frequently.

Software Defined Networking (SDN)/Network Function Virtualization (NFV) based caching scheme is proposed in [15] for future mobile networks. The solution is well defined for wireless networks but the authors do not incorporate end-to-end encryption in their solution.

Similar solutions for securing cached contents are proposed in [16], [17]. All these solutions do not seem to be practical for IoT devices having limited resources.

VI. Discussion

A. Privacy Issue

Convergent encryption is a well-known cryptosystem that is utilized to efficiently store duplicate files. However, utilizing this scheme to ensure end-to-end encryption in cached contents can raise privacy issues. The caching server can predict the nature of the content by observing encrypted chunks, which could easily be generated from a suspected list of files. To overcome this, the encrypted chunks and hash codes could be

masked with random numbers. These random numbers must be shared with the client out-of-band, along with the hash code and encrypted chunk ID.

B. Caching Location for Efficient Retrieval

The decision on where to cache the content along with the path to the client depends on the requirement of the application the end user is utilizing. If multiple users, in the proximity of a LAN, are accessing the same video from a content provider, the content provider can cache this video in the same LAN instead of caching it to a CDN server or even in a 5G node. The applications, which require ultra low latency can be served by caching content in a D2D node *e.g.,* a smartphone or any other computationally capable node within user's premises.

C. Business Model

It is important to note that although the caching decision is taken by the content provider, the other stakeholders involved in providing the content to the client, such as D2D nodes, cloudlet, IoT gateway, and 5G intermediate nodes must obtain some monetary benefits in providing their resources/services as a caching server. In this regard, a business model must be investigated that efficiently transforms the physical resources of a caching server into potential incentives it can gain from the content provider or the end user, who needs low latency and high throughput.

As a starting point, the business model could be along the lines of CDNs with an ability to evolve with the requirements of end users. For instance, in case of caching at D2D nodes, which are already limited in resources (*i.e.,* battery, processing, memory, storage *etc.*), D2D nodes can offer their resources only if they are rewarded with some incentives from the content provider or the end user. The Mobile Network Operator (MNO) can also be included in the business model even if the content is cached within user's premises (such as D2D nodes or cloudlets) since it can significantly save bandwidth of the MNO in both access and core networks.

VII. Conclusions and Future Work

In this paper, we present a marketplace for secure caching of the contents in 5G networks. We measure the throughput and latency gains for different file sizes at various locations as caching possibilities. We divide each file into smaller chunks for the possibility of reducing the duplicate contents at the caching servers. We argue that reducing duplicate contents not only provides the storage gain at the caching servers but also reduces the communication load over the links between caching servers and the content provider. Moreover, our simulation results demonstrate that caching contents near the end users, such as at the cloudlet enables fast delivery of the contents with significant throughput gains. In addition, we also find that dividing a file into larger chunks provides higher throughput gains as compared to smaller chunks.

In future, we aim to present more detailed analysis of our proposal by providing combined storage and security gain over other out-of-band caching schemes, such as blind cache. In addition, we aim to investigate more encryption schemes that can comply with the possibility of removing duplicate contents while preserving privacy at all ends.

Acknowledgements

This publication was made possible by National Priorities Research Program (NPRP) under grant number # NPRP7-125-2-061 from the Qatar National Research Fund (QNRF) (a member of Qatar Foundation (QF)). The statements made herein are solely the responsibility of the authors.

References

[1] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, May 2017.

[2] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, Aug 2014.

[3] M. K. Kiskani and H. R. Sadjadpour, "Secure coded caching in wireless ad hoc networks," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, Jan 2017, pp. 387–391.

[4] M. Usman, M. R. Asghar, I. S. Ansari, and F. Granelli, "Towards bootstrapping trust in D2D using PGP and reputation mechanism," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[5] G. Eriksson, J. Mattsson, N. Mitra, and Z. Sarker, "Blind cache: a solution to content delivery challenges in an all-encrypted web," *Ericsson white paper*, 2016.

[6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[7] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*. IEEE, 2002, pp. 617–624.

[8] R. Radescu and S. Pasca, "Experimental results in prediction by partial matching and star transformation applied in lossless compression of text files," in *2017 10th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, March 2017, pp. 17–22.

[9] N. Kozhemiakina, V. V. Lukin, N. N. Ponomarenko, J. Astola, and K. O. Egiazarian, "JPEG compression with recursive group coding," *Electronic Imaging*, vol. 2016, no. 15, pp. 1–6, 2016.

[10] D. Rebellion, "Video space calculator," last accessed: September 2017. [Online]. Available: https://www.digitalrebellion.com/webapps/videocalc

[11] J. Leguay, G. S. Paschos, E. Quaglia, and B. Smyth, "Cryptocache: Network caching with confidentiality," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[12] M. Mosko and C. A. Wood, "Secure off-path replication in content-centric networks," in *IEEE ICC 2017 Next Generation Networking and Internet Symposium (NGNI 2017). IEEE*, 2017.

[13] X. Yuan, X. Wang, J. Wang, Y. Chu, C. Wang, J. Wang, M. J. Montpetit, and S. Liu, "Enabling secure and efficient video delivery through encrypted in-network caching," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2077–2090, Aug 2016.

[14] F. Engelmann and P. Elia, "A content-delivery protocol, exploiting the privacy benefits of coded caching," in *2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2017, pp. 1–6.

[15] Y. Liu, J. C. Point, K. V. Katsaros, V. Glykantzis, M. S. Siddiqui, and E. Escalona, "SDN/NFV based caching solution for future mobile network (5G)," in *Networks and Communications (EuCNC), 2017 European Conference on*. IEEE, 2017, pp. 1–5.

[16] A. Sengupta, R. Tandon, and T. C. Clancy, "Fundamental limits of caching with secure delivery," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 355–370, Feb 2015.

[17] K. Thakker, C. H. Lung, and P. Morde, "Secure and optimal content-centric networking caching design," in *2015 Second International Conference on Trustworthy Systems and Their Applications*, July 2015, pp. 36–43.