Coded Caching Techniques: A Survey

by Yannick Klose

10. March 2023

Abstract

With the ever-increasing number of network requests and users on the internet caching has become an indispensable tool to reduce throughput and latency. In traditional caching the user stores some data locally in its cache, referred as the *local caching gain*. With more and more users and requests, optimising the load at the local side is no longer sufficient. Coded caching on the other hand makes use of the multicasting gain, creating a *global caching gain*, without cooperation among the users. With optimising both the placement phase and the delivery phase coded caching is able to serve multiple users with different request over fewer messages, reducing the load in the delivery phase dramatically. However with the introduction of coded caching techniques new challenges need to be solved. In this paper, we address the limitations and challenges of coded caching. For this purpose, we identify the three domains. First we compare methods to apply coded caching to *real world networks* architectures in the domain of mobile and static network structures and the resulting effects on both coding delay and transmission rate. Second we discuss the *user specific behaviour* and effects on the proposed coding schemes. Finally, we evaluate current *privacy methods* in coded caching and compare the load to non-private coded caching.

Contents

Chapter 1

Introduction

With the rising demands of more devices and users being connected to the internet and exchange data there is a need for techniques to further improve the requests times and provide an overall better user experience. A common technique is the concept of caching. The main goal of caching is to shift the traffic from peak hours, where resources are busy, to the off-hours, where resources are idle. This is done in two phases, called: *placement phase* and *delivery phase*. The placement phase describes the phase where popular content is stored on the local cache of the user. This phase happens during off-hours, where resources are idle. In the delivery phase, where resources are mostly busy, the user only need to requests data from the server that is not in the cache. This, depending on the caching strategy, can result in fewer requests to the server. While this traditional uncoded caching showed a significant improvement compared to networks that do not use caching there is still room for improvement.

In traditional caching the user stores the data on its local cache, called *local-caching gain*, making it highly dependent on the overall local cache size M. The lower the local cache size, the less significant traditional uncoded caching becomes. Assuming that a single server is responsible for K users that can in total store M (local caching) of total N files, the overall data-rate R in the delivery phase is described by:

$$R = K(1 - \frac{M}{N}) \tag{1.1}$$

Looking at 1.1, it becomes clear that with an increasing number of users and/or a high ratio of cache size M to the total number of files N, the required data R rate in the delivery phase strongly increases. As the amount of data available to users continues to grow and the number of users steadily increases, traditional caching is no longer an adequate solution.

In recent years many researchers have come up with new techniques, trying to lower the overall data-rate. Most of these techniques used mainly focussed on the users perspective, by analysing demands and trying to predict requests [1]. In 2014 Maddah-Ali et al. [2] introduced a new technique called: coded caching, that has gotten a lot of attention since then. Besides the local caching gain, coded caching introduces a *global caching gain* by jointly optimising both the placement and delivery phase. This enables coded caching to process request from different users within a single coded multicast transmission. The overall data-rate R can be described by the following term:

$$R = K(1 - \frac{M}{N}) \cdot \frac{1}{1 - K\frac{M}{N}} \tag{1.2}$$

This promising technique shows theoretic improvements in the delivery phase by a factor of up to 12 for any configuration of M, K and N. An example of a configuration with $K = 30$ and $N = 30$ for different rates of M is shown in figure 1.1

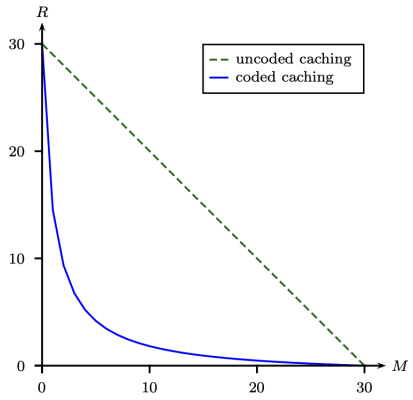


Figure 1.1 – Rate R required in the delivery phase as a function of memory size M for N = 30 files and K = 30 users. The figure compares the performance of the proposed coded caching scheme too that of conventional uncoded caching. (Figure used from Maddah-Ali et al. [2], page 3, figure 2)

With this general new concept comes many new challenges. One of these questions is which network architectures are realistic in real world environments. Can such models be also applied to mobile networks? How does coded caching handle many users, random activity and parallel computing? Does coded caching violates the privacy demands by using shared information? And if so, how can this issue be solved?

This paper gives an overview over the current coded caching techniques, describes the challenges and compares results from various papers. The paper discusses the problems of optimisation of placement and delivery phases, network architectures and privacy demand.

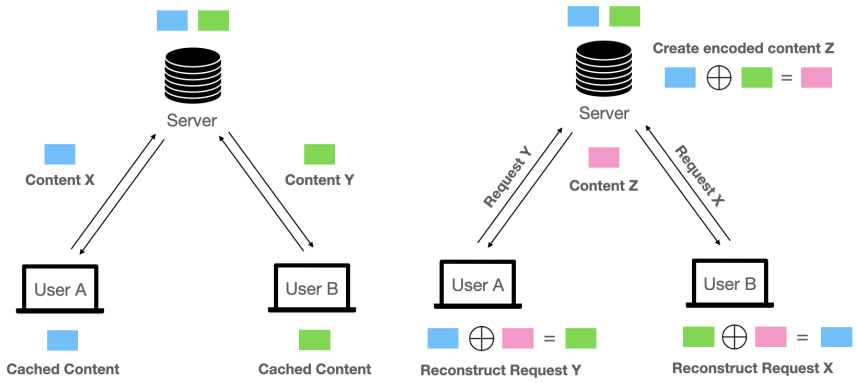
Chapter 2

Concept

The concept of coding caching was introduced by Maddah-Ali et al. [2] and describes a scenario in which a single server connected to through an error-free link to K users. There the server has access to a total number of N files $(W_1, W_2, ..., W_N)$ of size F in a database. Each user k on the other hand has its own isolated local cache Z_k of size MF, where $M \in [0, N]$ for some real number.

During the placement phase the users have access to a database of N files. Each user fills up its cache Z_k. During the delivery phase the users do not have access to database, instead each user k requests one file W_{d_k} to the server. The server then transmits a packet $X_{(d_1,...,d_k)}$ of size RF for some fixed real number of R. Given the local cache Z_k and the transmitted signal $X_{(d_1,...,d_k)}$, each user should be able to reconstruct their requested file W_{d_k}. The reconstruction can be described by a memory-rate pair (M, R), specifying the needed memory M of each user and the total rate R for the reconstruction. A memory-rate pair is said to be achievable if each user is able to reconstruct their request $d_1, d_2, ..., d_N$. $R^*(M)$ denotes the memory-rate-tradeoff, meaning the minimum required rate R for an achievable memory-rate pair. Maddah-Ali et al. [2] aims to find the minimum data-rate, that is achievable, for this specific scenario. A schematic overview of the process is shown in figure 2.1. On the left side the placement phase in shown in which each user stores some data locally. On the right side the delivery phase is shown in which the server fist creates the content Z, by XORing both requests and sending it over a multicast link. At the user-side, each user is able to reconstruct their requested file, since one pair is already known.

The proposed algorithm from Maddah-Ali et al. [2] can be seen in Algorithm 2.1. Within the placement phase two variables are defined: t and Γ. t indicates the ratio between M, K and N and is therefore an integer between 0 and K. Γ indicates the subset of users, that use the same cache configuration. For the case of $M = 0$ it is obvious that the server can simply transmit the union of the requested files. The



(a) placement phase **(b)** delivery phase

Figure 2.1 – Simple example of a placement and delivery phase for K=N=2 using coded caching.

overall minimum-rate-tradeoff $R^*(0)$ for this case is:

$$R^*(0) = F \cdot \min(N, K) \tag{2.1}$$

On the other hand for the case of $M = N$, the server does not have to transmit a single file, since the all data can be stored on the local cache Z_k by the user, resulting in:

$$R^*(N) = 0 \tag{2.2}$$

The interesting part begins when $0 < M < N$, such that $t \in [1, 2, ..., K-1]$. For that each file is split into $\binom{K}{t}$ subfiles of equal size (line 4 in 2.1). Each subfile $W_{n,\tau}$ is then stored in the users cache Z_k if $k \in \tau$, resulting in a total number of $N \cdot \binom{K-1}{t-1}$ subfiles for each user k. This also means that every t user store the same subfile in their cache. The total memory constrained of FM is still satisfied.

Next comes the delivery phase. In this phase the each user k request a file W_{d_k}, resulting in a total request vector of $(d_1, d_2, ..., d_N)$. Due to the described placement phase, we can split the users into two groups for each subfile that is requested. Let us define the subset $S \subset [K]$ with $|S| = t + 1$. If a user $s \in S$ requests a file $W_{d_s, S\setminus\{s\}}$, note that this file is present for every user $k \in S\setminus\{s\}$. Therefore, the server has to transmit the XOR signal of the requested files:

$$\oplus_{s \in S} W_{d_s, S\setminus\{s\}} \tag{2.3}$$

This can be applied to every subset of S. The resulting transmitted signal is defined in line 1 in 2.1 by $X_{(d_1, d_2, ..., d_N)}$.

Require: K users with local cache Z_k
Placement$(W_1, ..., W_N)$
$t \leftarrow MK/N$
$\Gamma = \{\gamma \subset [K] : |\gamma| = t\}$
for $n \in [N]$ **do**
  split W_n into $(W_{n,\gamma} : \gamma \in \Gamma)$
end for
for $k \in [K]$ **do**
  $Z_k \leftarrow (W_{n,\gamma} : n \in [N], \gamma \in \Gamma, k \in \gamma)$
end for
Delivery$(W_1, ..., W_N, d_1, ..., d_K)$
$t \leftarrow MK/N$
$\sigma \leftarrow \{S \subset [K] : |S| = t + 1\}$
$X_{(d_1,...,d_K)} \leftarrow (\oplus_{k \in S} W_{d_k, S\setminus\{k\}} : S \in \sigma)$

Algorithm 2.1 – Coded Caching Algorithm proposed by Maddah-Ali et al. [2]

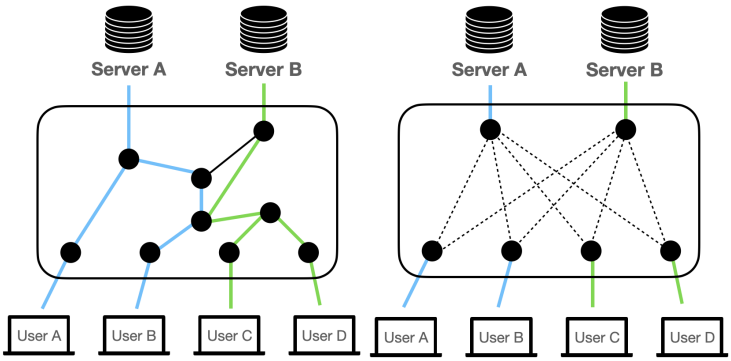
Maddah-Ali et al. [2] impressively showed that this technique is able to reduce the data-rate in the delivery phase by a big margin compared to traditional uncoded caching. In Figure 1.1 we can see the comparison of the proposed caching technique compared to traditional caching. While the extreme cases are the same for both techniques the described case of $0 < M < N$ outperforms the traditional uncoded caching.

Network Structures in Coded Caching

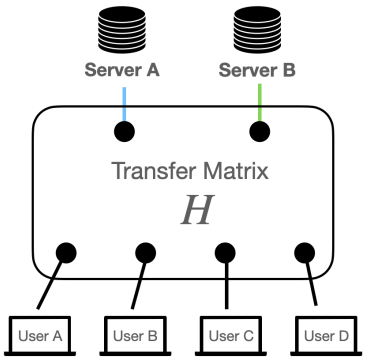
Shariatpanahi et al. [3] considered a multi-server coded caching technique with 3 different network structures. These network structures were defined as: *dedicated networks*, *flexible networks* and *linear networks*. The different network structures are shown in Figure 2.2. Each of the network structure involves multiple servers that are used for coded caching. The network is defined as a Directed Acyclic Graph (DAG) $\mathcal{G} = (V, E)$ where V are the vertices of the internal nodes and $e \in E$ the *error-free* and *delay-free* edges that have a capacity of one symbol per channel.

In Dedicated networks each server is connected through a fixed link to the users. This means each server is responsible for a non-overlapping fixed subset of users. The number of users each server is responsible for is distributed equally over the total number of users. With respect to load balancing Shariatpanahi et al. [3] therefore proposes that each Server L is responsible for K/L users. Using that simple and naive approach the overall delay in the network can be described as follows:

$$D^*(M) \leq K'(1 - \frac{M}{N}) \cdot \frac{1}{\min(K', L + K'\frac{M}{N})} \cdot \frac{F}{m} \tag{2.4}$$



(a) Dedicated Network **(b)** Flexible Network

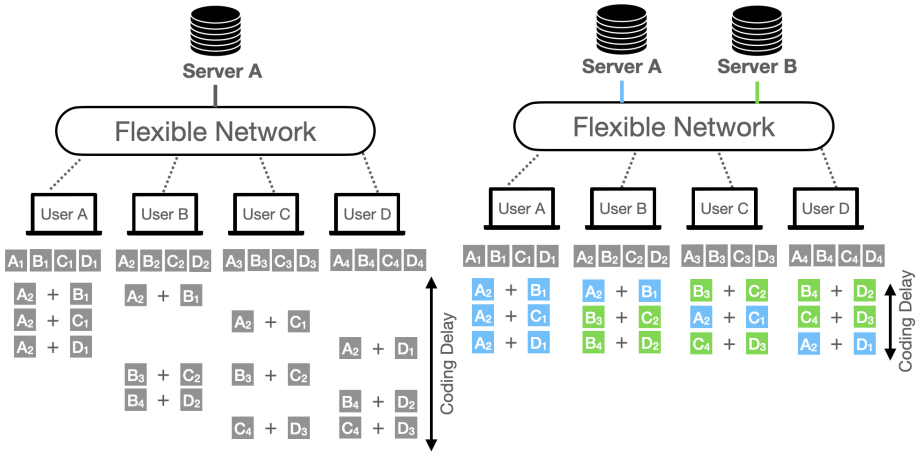


(c) Linear Network

Figure 2.2 – Three network architectures used by Shariatpanahi et al. [3] (compare figure 2, page 4)

In Equation 2.4 K' indicates the smallest number larger or equal to K which is divisible by L. Since in this case we are looking at the upper bound of the delay and not the overall data-rate, we have to multiply $\frac{F}{m}$, which indicates the ratio of the file-size and the m-bit Symbol. Comparing this to the original data-rate proposed by Maddah-Ali et al. [2], the data-load is distributed among the L server that can operate in parallel. This results in an overall lower data-rate $R^*(M)$ and therefore also lower network delay $D^*(M)$. If K' is not divisible by L, virtual users are added, to satisfy the constraint [3].

An improved upper bound of the network delay can be achieved by using flexible networks. Flexible networks are similar to dedicated networks but allow a higher degree of freedom with respect to the user assignment. Instead of fixed user subsets, flexible networks can arbitrarily change the assignment of users for each transmission, allowing parallel operations of the servers. To operate in parallel each server can have a different strategy p to transmit the corresponding parts to each user. In a simple scenario of two servers operating in parallel, satisfying 4 users is shown in



(a) One Server **(b)** Two Server (parallel)

Figure 2.3 – Concept of reducing the total coding delay in flexible networks used by Shariatpanahi et al. [3] (compare figure 5, page 8)

Figure 2.3. We can clearly see the two strategies (blue and green) that overall reduce the network delay. The overall delay T_C is derived in 2.5 and can be summarised for each server i with strategy p_i as:

$$T_C = \frac{F}{m} \cdot \frac{1}{\sum_{i=1}^{L} \frac{p_i}{K-p_i+1}} \tag{2.5}$$

While the result in 2.5 shows a decrement of the overall network delay, flexible and topological complex networks require a proper routing strategy. This means that the topology of the network must be known by the nodes. However, this is often not the case. Therefore Shariatpanahi et al. [3] introduces linear networks that performs simple random linear network coding, which was already used in the work of Das et al. [4]. While simple random linear network coding may not result in an optimal solution, Shariatpanahi et al. [3] consider this method as a more realistic architecture in real world networks. The overall upper bound of the delay for linear networks is [3]:

$$D^*(M) \leq K(1-\frac{M}{N}) \cdot \frac{1}{\min(K, L+K\frac{M}{N})} \cdot \frac{F}{m} \tag{2.6}$$

Similar to the approach by Shariatpanahi et al. [3], Bayat et al. [5] also criticises the network structure from the original proposed scheme [2]. While Maddah-Ali et al. [2] assumed the network structure was fixed to a simple server-client configuration, Shariatpanahi et al. [3] showed three different networks mentioning that finding an optimal routing solution can be hard. Bayat et al. [5] however present an algorithm that formulates the optimal routing scheme as an optimisation problem. With this

solution the authors claim to solve problems such as asynchronous streaming sessions, finite file size, scalability of the scheme to large and spatially distributed networks, user mobility, random activity and many more.

Bayat et al. [5] argues that the standard approach of simulating networks is not realistic. In mobile networks for example a centrally coordinated prefetching phase would be very impractical, since all users would have to be connected to a giant antenna. Furthermore, such networks have to deal with random activity, meaning users might join and leave the network randomly. Therefore Bayat et al. [5] proposes to use *decentralised* prefetching schemes.

The overall proposed network structure consists of one server and many helper nodes H. Based on the geographic location each user k is connected to a nearby helper node. The overall network structure is seen in Figure 2.4.

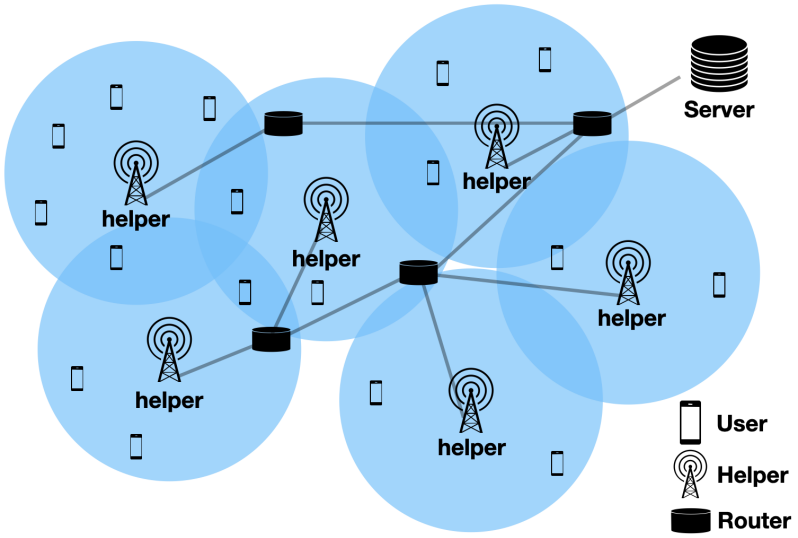


Figure 2.4 – A schematic overview of a mobile network with one server, multiple helpers and connected users. The routing is implemented with coded caching via a multicast IP network. (compare Bayat et al. [5] figure 1, page 5)

Bayat et al. [5] proposes a new delivery scheme based on routing optimisation. Instead of having one cache configuration of a single server like Shariatpanahi et al. [3] , Bayat et al.[5] uses virtual users L to divide the W_i into $\binom{L}{t'}$ pieces, where $t' = \frac{L \cdot M}{N}$ is the library replication parameter. Furthermore, they do not use multiple servers, but a decentralised prefetching phase where each user k that joins the network picks at random a configuration $l \in [L]$ that will be loaded in its cache Z_l. Common techniques such as multi-route delivery scheme proposed by Parrinello et al. [6] do not work well on such topological networks as described in Figure 2.4, because the users with the same cache configuration are not equivalent, because of the network topology.

Another problem, especially in real world mobile networks, is interference [5]. Simply scaling the number of helpers to balance the load does not work, due to interference. An example of a possible real world scenario in mobile networks is shown in figure 2.4, where multiple nodes overlap. One common technique to avoid interference is to use different frequency bands. However, finding the best solution for such a problem is computational complex. One way of reducing the complexity is using a greedy algorithm where the complexity is upper bounded by $L \cdot \log(L) \cdot K$. In this approach first each unique user is uniquely assigned and all other users are assigned one-by-one in such a way that the overall total objective function is minimised.

The second suggestion by Bayat et al. [5] is using the so-called *avalanche* scheme. This scheme is a routing and scheduling strategy that takes collisions into account and resolves these over multiple time slots. This yields to an overall better worst-case delivery time. In this algorithm the users are divided into two groups, the ones that experience interference, and the ones that do not. The overall reuse factor for all users in the avalanche scheme is 1. To indicate sets of non-interference users that are uniquely assigned to the helper h with caching group l, we use P_l^h. Compared to other reuse schemes, the avalanche scheme is able to dynamically adjust the delivery list L^h, which is the union of the non-interference users P_l^h. The algorithm can be expressed as follows.

1: **Initialisation:** Set all users to unserved,
2: **while** $U_{unserved} \neq \emptyset$ **do**
3: **Find helpers finishing in column:** such that time Δ_{min} is minimized
4: **Increment time:** $D+ = \Delta_{min}$
5: **Update unserved users**
6: **Identify stopping helpers:** all helpers where $L^h = \emptyset$
7: **Update delivery arrays of active helpers:** add unserved users k to L^h
8: **Reactivate helpers**
9: **end while**

Algorithm 2.2 – Avalanche Algorithm proposed by Bayat et al. [5]

The results using the 2.2 shows an algorithm that takes inference of nodes into consideration while being able to minimise the overall delivery time compared to traditional multi-route algorithms Parrinello et al. [6]. The proposed avalanche scheme shares many similarities with the CSMA-based random access.

User Behaviour

Another important property for coded caching is the user behaviour. In most coded caching techniques the file distribution is assumed to be equally distributed. However, for most practical applications this is not the case. Most users follow a non-uniform

distribution, meaning there is more popular content than other. Niesen et al. [7] shows that the approach that traditional caching techniques do, caching the most popular content, is rather suboptimal. One way of handling non-uniform file distribution is making the memory size M, during the placement phase, dependent on the file popularity. However, this breaks the symmetry in the placement phase. The scheme proposed by Niesen et al. [7] instead groups files with similar popularity content together. Thereby the symmetry in the placement phase is preserved.

Since another user behaviour is that popular content changes over time, Pedarsani et al. [8] proposes a scheme to update the cache in an efficient way, called *coded least-recently sent* (LRS). Pedarsani et al. [8] argues that the *least-recently used* (LRU), that was already proven in 1985 to be optimal by Sleator et al. [9], cannot be applied to coded caching in an efficient way, since now there is not a single cache anymore, but multiple caches. Pedarsani et al. [8] focusses on an online-caching approach. All files that have been partially cached, use the coded caching used by Maddah-Ali et al. [2], and for the ones that have not been cached uncoded caching. Then the user k replaces the least recently used file in its cache with a random subset of MF/N bits of the send file $d_t(k)$. Therefore if *any* user do not have the file cached, all users store such a random subset. The total number of partially cached files stays the same. Ji et al. [10] extended the work of Pedarsani et al. [8] by using a configuration of a single source (server) connected to n nodes with caches, with a total of m files that can request multiple files. The probability distribution $q = (q_1, ..., q_m)$ is assumed to be known a priori. Each request from the users is done independently from this distribution. The proposed caching strategy is called *Random Popularity-based (RAP)* Caching.

Privacy methods

Another important concept of caching deals about privacy preserving coded caching techniques. Since the introduction of coded caching by Maddah-Ali et al. [2] there has been attempts to preserve the privacy in the network. One of the first privacy preserving techniques came up by Sengupta et al. [11]. They proposed a scheme that secured the delivery scheme by using a technique similar to code-division-multiple-access (CDMA). Besides the caching content the user also stores a key that encrypts the multicast messages. Therefore, only the assigned user can decrypt the message. This method was proven to be very close to the non-secure case for large number of users and cache size.

However, the described privacy technique by Sengupta et al. [11] could be vulnerable by malicious users that perform data observation and learn patterns. Therefore, Engelmann et al. [12] suggests using L virtual users, so called phantom users, that randomly requests a file. The total number of users is therefore defined as

$K = K' + L$ where K' is the number of real users. The price of this anonymisation has to pay is an increase of load. This on the other hand is considered to be negligible when the number of K and K' users increase [12].

Wan et al. [13] argues that by adding virtual users that randomly request data, even if scaling the number of users to infinity this would not satisfy the information-theoretic privacy constraint. Instead Wan et al. [13] suggests using a scheme in which in total $\binom{N}{L \cdot K - K}$ virtual users are generated. Given that configuration each subset L is requested by exactly K users, thus hiding the total number of real users among the effective users.

Since the main limitation is the sub-packetization level that is $\mathcal{O}(2^{\binom{N}{L \cdot K}})$ Wan et al. [13] propose using the *MDS-based* (Maximum Distance Separable) scheme. In this scheme each file F_i is split into multiple Blocks B and then encoded by a MDS code. This allows each file to be reconstructed by any MDS coded symbols. Overall, the authors claim, that this approach provides full information-theoretic privacy of the users.

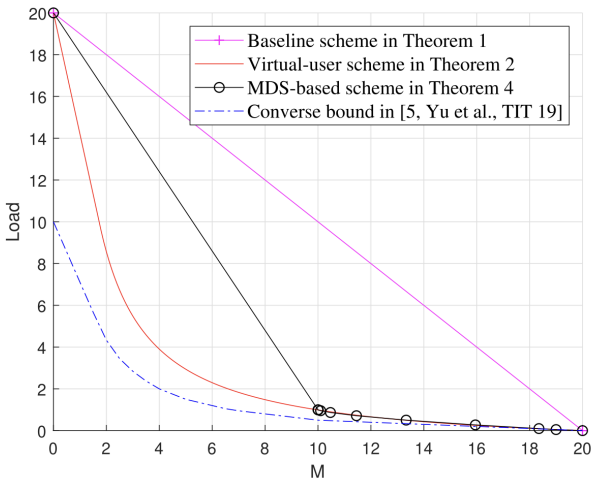


Figure 2.5 – Comparison of private coded caching techniques used by Wan et al. [13] in figure 2.

In figure 2.5 the two proposed private coded caching techniques from Wan et al. [13] are shown and compared with non-privacy coded caching techniques, described as the converse bound from Yu et al. [14]. Both of the two proposed methods show a lower load than the baseline, using no coded caching. Wan et al. [13] showed that for $M < N/2$ the load for the virtual user scheme is lower than the MDS-based scheme. While for $M \geq N/2$, both schemes are similar.

Chapter 3

Discussion

Coded caching introduces a new way of rethinking the traditional caching approach. With these new techniques new questions arise, whether the algorithms and experiments reflect real-world scenarios and satisfy the needs of the users. The key limitation of traditional caching is the overall limited data-rate due to a limitation of resources. Serving users during off-hours where resources are idle is therefore not a problem, that caching addresses. A naive approach to overcome this delay users have to face during peak-hours is to extend the available resources, by adding more servers and a higher connectivity in the network. However, this approach is neither good in an economical perspective nor a sustainable solution to the problem. Coded caching on the other hand introduces this new way of using the same resources but way more efficiently. To summarise the newly occurred challenges and important aspects in coded caching we want to create three categories. The first subsection discusses coded caching with respect to real world networks. In the second subsection the user behaviour is analysed and compared to the different approaches in coded caching. Finally, privacy concerns are discussed. An overview of the different domains is shown in Figure 3.1

Real World Networks

While Maddah-Ali et al. [2] have set the first milestone for a completely new concept of caching, there are still some points of criticism or further development, some of which partly have been pointed out by the authors themselves. First, they assumes a static network architecture of a single server, connected through K users, which is clearly not a realistic scenario for a network structure in a real world scenario, according to Shariatpanahi et al. [3]. Real world scenarios include different network architectures [3], shown in figure 2.2. The linear network is one way of approximating the optimal solution because in most network structures the architecture is not

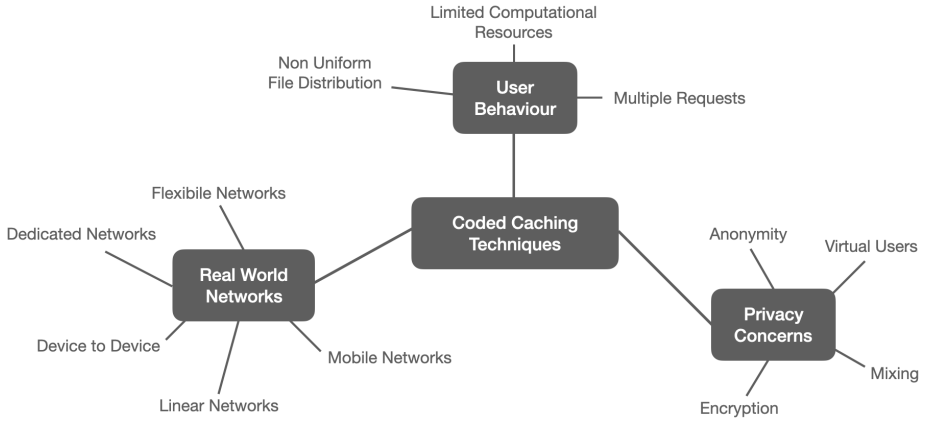


Figure 3.1 – Domains in Coded Caching Techniques

well known. While there are some conditions where the network structure is not well known, one can still consider some general properties about real word networks. In the past many real world networks have been considered as random graphs, while Watts and Strogatz first compared them to real world networks. Therefore, we consider an analysis with respect to average path length and clustering coefficient an important comparison that is not used by Shariatpanahi et al.[3]. A comparison between network structures that use small world properties (with small diameter and high clustering coefficient) and Scale-Free Networks where most nodes have very low degrees and few central nodes a high degree would be an interesting complement in our opinion. When analysing real world networks an important metric is the robustness of the networks. We agree that the approaches should differ for different applications. When looking at wireless communication e.g. in mobile networks one has to consider the interference of nodes and basestations, as described by Bayat et al. [5]. Both Shariatpanahi et al. [3] and Bayat et al. [5] suggest using multiple servers to make use of parallel processes and overall reduce the request time during the delivery phase. What is not considered is the use of edge computing for the coded caching helper nodes. This could lead to an overall lower bound compared to the one in Shariatpanahi et al. [3]. Furthermore, when using decentralised caching approaches one has to think about incentives for the users of caching content. While this has been discussed with respect to traditional caching by Chen et al. [15], this must be reviewed for coded caching. Overall, some challenges regarding general network structures in Maddah-Ali et al. [16] could be solved with the work from Bayat et al. [5], however, extensive evaluations of different real world environments still have to be done.

User Behaviour

Another important aspect that should be discussed is the behaviour of users. Some mathematical theorems [2], [3] assume a uniform file distribution. However, in our option this is not something that reflects the users behaviour. Instead, file contents follow a popularity distribution, meaning for short amount of times there is a very high request while after a certain time period there are almost no requests anymore. Various works [10], [8], [7] show a technique how the placement phase should be adapted in coded caching in order to achieve a better performance, compared to traditional coded caching. However, in this technique the popularity distribution is assumed to be known a priori, which is something we do not consider to be realistic. Bharath et al. [17] uses a learning-based approach in order to estimate such distribution. Another important constraint for good user experience is the delay constrained. As suggested by Bayat et al. [5] each large file, e.g. video file, can be divided into blocks and then be processed. However, what is not considered during the results is the impact of other constrains such as the burden of computational cost, since each user has to reconstruct their request from the given message X_d and their local cache Z_k. Following this, an additional metric should be the total energy consumption since this is crucial, especially when considering mobile users.

Most of the proposed methods use optimisation strategies in order to solve predefined quality of service (QoS) metrics. However, we do not consider this as the only good option in order to solve these problems. Especially when looking at cellular networks, approaches such as Game-Theoretic analysis and data driven approaches such as Machine Learning could be used in order to optimise the routing strategies based on user behaviour, similar to what Li et al. [18] suggests. Again, these methods require consistent and representative user data on real networks to produce credible results, which is currently not the case. Datasets such as the one from Netflix, used by Pedarsani et al. [8], must be used as standard comparison throughout different coded caching approaches.

Privacy Concerns

Since the core principle of coded caching lays in the concept of the global caching gain, meaning multiple users can satisfy their request by one multicast transmission, privacy concerns came up very quickly and firstly described by Sengupta et al. [11]. While they describe the idea of using a key to encrypt the results of the multicast transmission this in our option only solves one part of having a secure protocol.

When it comes to securing traffic on the internet, there are many approaches that define security into different categories like: confidentiality, integrity, availability, authenticity, non-repudiation, anonymity and many more. These security goals in our option can also be applied to coded caching techniques. The difficulty is that

different to traditional data exchange in coded caching the data has to be reused for multiple users. While Sengupta et al. [11] cover the goal of the confidentiality by using encryption, the anonymity is not satisfied by any means. Simple traffic analysis can reveal the users behaviour, even if the server or an eavesdropper only sees the pseudonyms. In internet security such probability of linking users with requests can be reduced by methods such as mixing networks or layered encryption as it is done in onion routing. The method proposed by Engelmann et al. [12] where phantom users are added to the original set of real users shows similarities to pooled mixing principles. However, for mixing techniques there exists also adversity models that in our option can be applied as well to the proposed technique of [12], e.g. the blending attack, where the attacker degrades the anonymity set by knowing the requests from the users. Furthermore, the security measurements especially in peer-to-peer networks require extra headers to store keys and therefore reducing the efficiency per request.

However, all the proposed privacy methods [13], [12], [11] do not consider the privacy aspect in the placement phase and rather focus on the delivery phase. This can be considered as a major issue since given the information from the placement phase an adversary could take inferences to the delivery phase. Overall, many parallels can be drawn from existing knowledge in the field of privacy, yet practice shows that many methods have weaknesses in the course that can only be found through extensive testing.

Chapter 4

Conclusion

In this article we have seen many techniques and concept of current of coded caching algorithms. We observed that there are three main domains that need to be considered when designing a good coded caching technique. First the algorithm has to be adequate for the used network architecture. Second the algorithm must consider a specific user behaviour. Finally, all these needs have to be executed under privacy measurements. Thereby the needs for these domains can differ from the situation. Some concepts consider mobile networks, some flexible networks. When it comes to user behaviour users can randomly join and leave in the network, do multiple requests, and use non-uniform file distribution due to popular content. Regarding the privacy demands the biggest challenge is to reduce the overhead for header or extra data to manage keys, etc. while still reducing the overall request time. The metrics used are almost exclusively related to the load in the delivery phase in terms of coding delay and throughput. However, for realistic comparisons, other metrics must be considered, such as the computing effort for reconstruction, the associated energy consumption, the failure rates, and much more. In order to validate these results there is a need for large standardised tests on real world data. Further research will therefore deal with stimulation of real networks, optimisation of placement and delivery phase based on real user specific behaviour and new methods for privacy compliance.

List of Figures

Bibliography

[1] A. Meyerson, K. Munagala, and S. Plotkin, "Web caching using access statistics," *Society for Industrial and Applied Mathematics*, pp. 354–363, 2001.

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental Limits of Caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[3] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-Server Coded Caching," *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 7253–7271, 2016.

[4] A. Das, S. Vishwanath, S. Jafar, and A. Markopoulou, "Network coding for multiple unicasts: An interference alignment approach," *IEEE International Symposium on Information Theory*, pp. 1878–1882, 2010.

[5] M. Bayat, K. Wan, and G. Caire, "Coded Caching Over Multicast Routing Networks," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3614–3627, 2021.

[6] E. Parrinello, A. Ünsal, and P. Elia, "Optimal coded caching in heterogeneous networks with uncoded prefetching," *IEEE Information Theory Workshop*, pp. 1–5, 2018.

[7] U. Niesen and M. A. Maddah-Ali, "Coded Caching With Nonuniform Demands," *IEEE Transactions on Information Theory*, vol. 63, no. 2, pp. 1146–1158, 2017.

[8] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online Coded Caching," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 836–845, 2016.

[9] D. D. Sleator and R. E. Tarjan, "Amortized Efficiency of List Update and Paging Rules," *Communications of the ACM*, vol. 28, no. 2, pp. 202–208, 1985.

[10] M. Ji, A. Tulino, J. Llorca, and G. Caire, "Caching-Aided Coded Multicasting with Multiple Random Requests," *IEEE Information Theory Workshop*, pp. 1–5, 2015.

[11] 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, 2015.

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

[13] K. Wan and G. Caire, "On Coded Caching With Private Demands," *IEEE Transactions on Information Theory*, vol. 67, no. 1, pp. 358–372, 2021.

[14] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 647–663, 2017.

[15] Z. Chen, Y. Liu, B. Zhou, and M. Tao, "Caching incentive design in wireless D2D networks: A Stackelberg game approach," *International Conference on Communications*, pp. 1–6, 2016.

[16] M. A. Maddah-Ali and U. Niesen, "Coding for caching: fundamental limits and practical challenges," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 23–29, 2016.

[17] B. N. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Transactions on Communications*, vol. 64, no. 4, pp. 1674–1686, 2016.

[18] L. Li, G. Zhao, and R. S. Blum, "A Survey of Caching Techniques in Cellular Networks: Research Issues and Challenges in Content Placement and Delivery Strategies," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 1710–1732, 2018.