Evaluating the Latency Overhead of Network-Coded Cooperative Networks for Different Cloud Sizes

Roberto Torre*, Hani Salah*, Giang T. Nguyen*, and Frank H. P. Fitzek†

*Deutsche Telekom Chair of Communication Networks, Technische Universität Dresden, 01062 Dresden, Germany
Email: {firstname}.{lastname}@tu-dresden.de
†Centre for Tactile Internet with Human-in-the-loop (CeTi), Technische Universität Dresden, 01062 Dresden, Germany

Abstract—5G communication networks introduce a massive heterogeneous environment where network traffic mainly contains video data. Today, in situations where users are closely located to each other and request the same information (e.g., streaming services in stadiums, connecting information in trains, slides dissemination in conferences, etc), the base station (BS) establishes one unicast connection per user, which is inefficient since the BS sends repeated data to co-located users. Network-Coded Cooperative (NCC) networks increase the network performance in the aforementioned scenarios by leveraging the use of Random Linear Network Coding (RLNC) and Mobile Clouds (MC). Non of the previous works evaluated the latency overhead of NCC systems, to the best of our knowledge. Our contribution in this paper is twofold: First, we present a new method to measure the latency where clocks inside the machines do not need to be synchronized. Second, we use that method and the traditional timestamps to measure the latency in a NCC testbed. The results show an average latency of 200 milliseconds (ms) for three clients and 400 ms for four clients. Moreover, we observed a correlation between clients inside the MC in terms of packet loss. A loss would stall the packet decoding until it is recovered, pictured by latency peaks.

Index Terms—Network-coded cooperation, data dissemination, network coding, random linear network coding, mobile clouds, latency evaluation

I. INTRODUCTION

As a new mobile generation unfolds, cellular networks are undergoing a major shift in their deployment and optimization. Renewed infrastructures, novel protocols, and new technologies are used in the next generation of mobile communications. 5G. Cisco [1] reported that mobile data traffic will increase sevenfold between 2017 and 2022, reaching 77.5 exabytes per month. Most of the mobile data traffic can be attributed to IP video traffic, which will represent 82% of all IP traffic. Immersive media services [2] will have a high impact in the following years. For instance, live video will grow 15-fold, video surveillance cameras traffic will increase by around 700%, and virtual reality (VR) and augmented reality (AR) traffic will reach a CAGR of 65% [1].

Currently, user equipments (UEs) that request access to a given content (e.g., video streaming) through LTE-A are connected via a unicast session from the cellular base station (eNB), regardless of the number of UEs that request the exact same content. In scenarios where multiple clients are closely located to each other, the establishment of replicated unicast sessions becomes inefficient. Examples of such scenarios include the audience in stands receiving a 360-degree video stream from the football match they are watching, passengers in a train who receive information from further connections, players of AR or VR games like Pokemon Go†.

The research community is aware that the current LTE-A system will not be able to handle the expected increase in data traffic in the upcoming years. Consequently, the 3GPP group developed a protocol to broadcast information in a cellular network, called Enhanced Multimedia Broadcast Multicast System (eMBMS). However, a European report detected several issues in a deployment test [3]. Some of those issues were spectral efficiency, energy consumption, and coverage. Consequently, new cellular dissemination methods are being designed. Today, one of the methods that provides a high throughput along with high resilience and low energy consumption is Network-Coded Cooperative (NCC) networking [4]. These networks leverage the interplay between Network Coding (NC) [5] and Mobile Clouds (MC) [6] to efficiently multicast information to co-located users. Despite the research done in NCC networks, there are still unknowns such as the number of cooperating users in the cloud, the number of redundancies the NC protocol must use, and the latency overhead produced by NCC.

This paper extends upon our previous works [7], [8]. In particular, in this paper, we aim to understand the latency overhead incurred by the NCC protocol for different amounts of clients in the cloud. Towards that end, we changed the testbed presented in [8] by adding a camera to the server. To evaluate the latency overhead, we propose a new method that takes snapshots of a clock and compares the difference between clocks. Finally, we take measurements of the proposed method and the conventional one, that is, using timestamps.

The evaluation results show an acceptable latency overhead for video streaming, AR, and VR services. They also show more sparse results in the four-client scenario. Moreover, we observe a clear correlation between errors in the clients. We observe the impact on latency when packets are lost for two different cases: (i) when packets are recovered by NC redundancies, and (ii) when packets are not recovered.

The remainder of the paper is organized as follows. First, we discuss the background and related work in Section II. Second, we describe the two methods used to measure the latency.

†Niantic Inc. Pokemon Go. https://www.pokemongo.com
in Section III. Afterwards, we present the settings that lead to the adequate configuration of the system, followed by the evaluation results in Section IV. Finally, Section V concludes the paper.

II. BACKGROUND AND RELATED WORK

A. Random Linear Network Coding

Random Linear Network Coding (RLNC) [9] is a forward error correction protocol that facilitates reliable data transfer as well as data storage. RLNC has been introduced to increase network throughput, efficiency, and resilience. NC [5] has been in the research community for more than 30 years and has been proved to increase network throughput and resilience [10].

One of the main features behind RLNC is that, instead of sending raw packets to the destination, they are grouped and mixed together (encoded) as a linear combination of packets. Hence, every coded packet has information from all the fellow packets in the group (called generation). The main strength of RLNC resides in its ability to recover from losses without need for acknowledgments or retransmissions. The destination, in order to be able to recover (decode) the information, only needs to receive enough linear independent packets to fill the coding matrix and extract the information via Gaussian elimination. The main benefit of using RLNC in this setup is that every NC coefficient is random, so during the recoding phase the UEs do not need to wait for any deterministic coefficient. Instead, they simply recode the incoming packet with a new random coefficient and change the RLNC header. This method provides better performance in lossy channels because instead of retransmitting lost packets, the encoder sends a couple of extra coded packets, known as redundancies. The coding ratio, denoted as $CR$, represents the density of coded packets per generation $g$. It is defined as the number of total packets (uncoded and redundancies) divided by the generation size. We express this number in percentage. Hence, we represent the coding ratio as $CR = (g + c)/g \cdot 100$, where $c$ is the number of redundancies per generation. High mobility scenarios where clients leave and arrive continuously are prone to high losses. In these scenarios, a random versatile code such as RLNC is likely to succeed more than deterministic codes like Reed-Solomon.

Ho et al. [9] were the first to introduce RLNC. They used a full-coded version of RLNC, where all packets were encoded and sent to the destination. This was later overperformed by Systematic RLNC [11], whose authors sent first the whole generation unencoded (systematic packets) and then full-coded redundancies, and thus, reduced the protocol complexity. Pace RLNC [12] introduced coded redundancies in between the systematic packets, to reduce the latency of error recovery. An improved version of Pace RLNC is Pace Multigeneration [13], a novel protocol that leverages the use of subdecoders to increase resilience to network jitter.

B. Network Coded Cooperative Relaying

As mentioned above, data dissemination in cellular networks is in the spotlight of the research community. The 3GPP group developed a system named eMBMS [14] to disseminate data to multiple users close to each other. Unfortunately, a European Technical Report reported several issues in terms of coverage, energy consumption, and spectrum [3], which opened up new challenges.

Apart from NC, Cooperative Relaying has been introduced to improve network performance on its own [15]. Grouping clients in MCs [6] increases network throughput and reduces energy consumption. The combination of both NC and Cooperative Relaying improves performance by leveraging the use of both technologies. MicroCast [16] used this technology to create a solid system with an integrated interface that efficiently streamed video. CoopStream [17] used the same system, but with the use of RLNC instead of basic NC. In both cases, the short-range communication was unicast. In cases where the number of clients could scale, the number of needed unicast sessions was huge. Hence, NCVCS [18] introduced the same principle but with the usage of multicast sessions in cooperative communication. However, this was not used in cellular networks, but only in local area networks (LANs). NCC [7], [8] networks address this issue, using multicast and RLNC in cellular networks. Table I collects the main approaches used in NCC Relaying and compares them with eMBMS.

<table>
<thead>
<tr>
<th></th>
<th>LTE-A</th>
<th>NC</th>
<th>Short-range</th>
<th>FEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMBMS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Raptor Codes</td>
</tr>
<tr>
<td>MicroCast</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Unicast NC</td>
</tr>
<tr>
<td>CoopStream</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>RLNC</td>
</tr>
<tr>
<td>NCVCS</td>
<td></td>
<td>✓</td>
<td>Multicast NC</td>
<td></td>
</tr>
<tr>
<td>NCC system</td>
<td>✓</td>
<td>✓</td>
<td>Multicast RLNC</td>
<td></td>
</tr>
</tbody>
</table>

C. Network-Coded Cooperation

In this subsection, we condense the main features of NCC, which were already presented in our previous work [7]. NCC extends the concept of secret small cell [20] to a more specific one. It defines a MC as a group of clients inside a small cell that shares their resources opportunistically, cooperating with each other to obtain a common benefit [6]. NCC uses MC to offload the traffic from the cellular network into a short-range network inside the small cell, like WiFi. The protocol comprises two different phases, the cellular phase and the cooperative phase, which can occur sequentially or in parallel.

The cloud size, $n$, represents the maximum number of clients that a MC can handle. There is no need to assume that every member in the MC has a cellular communication with the eNB. Nevertheless, it is the most common assumption by previous works [7], [8]. They all request the same content to the server and clients can eventually join or leave the cloud. MCs are formed in periodic formation phases, where clients can join or leave the cloud on the fly. Therefore, the structure of the MCs can be modified at each formation phase, but the maximum cloud size remains $n$. That is, clusters of $n' < n$...
UEs are only formed if the number of remaining UEs that request the same video stream is less than \( n \). MC formation phases are centralized, that is, the eNB is in charge of the formation of the MCs as it possesses a global view of the network.

**Cellular phase**: The eNB distributes the packets of the generation, hereafter denoted as \( g \), to the UEs connected via time-multiplexed unicast sessions in a round-robin fashion. Each of the \( n \) UEs is assigned an index, in the set \( N = \{ i \in \mathbb{Z}_+ \ | \ i \leq n \} \), that defines the order in which they will receive the data packets from the eNB. That is, as illustrated in the left part of Fig. 1, the first packet is sent to the first (black) UE, in the first time slot. In the second time slot, the eNB sends the second data packet to the second (blue) UE, and so on. The eNB will send another packet to the first UE in the \((n + 1)\)th time slot after sending a data packet to the \( n \)th UE; this will be the \((n + 1)\)th packet of the generation. At the end of this phase, all \( g \) packets will be distributed, and each UE will have \([g/n]\) or \([g/n] \) packets depending on the order the connection between the eNB and the UE was established.

**Cooperative phase**: When a UE receives a packet stream from the source, it will be in charge of redistributing the packet to the rest of the clients in the MC. Since no feedback messages are transmitted, the eNB must inform the number of time slots allocated for the content distribution within the MC to the UEs. Each UE will be assigned an index \( i \) in order to create the TDMA schedule in the cellular phase. At every time slot, each UE sends a WiFi multicast packet to the remaining UEs in the small cell, and the transmitting client is changed to distribute all resources in the small cell uniformly. The time slot in this phase does not need to be the same as in the cellular phase since a different data rate can be used.

A timing diagram of our NCC protocol is depicted in Fig. 1. In this diagram, we show how the protocol recovers from an error occurred in the second time slot with the first coded transmission. This example comprises 3 clients, a generation of 5 packets and 1 coded transmission.

**III. Measuring Latency**

In this section, we introduce the two methods that we propose to measure latency in the system. In the NCC scenario, we describe latency as the period of time taken between the video encoded in the server and the video decoded in the clients. To successfully evaluate the latency, the clocks in the server and clients need to be synchronized. This, however, is hard to achieve due to imperfections of the clock oscillator. Ideally, the clock should be \( C(t) = t \), where \( t \) denotes the reference time. In reality, the clock of a device is modeled as

\[
C(t) = \theta + f \cdot t
\]

where \( \theta \) in the clock offset and \( f \) is the clock skew, respectively. From this equation we can express the clock relationship between two devices (A and B) as

\[
C_B(t) = \theta^{AB} + f^{AB} \cdot C_A(t)
\]

where \( \theta^{AB} \) and \( f^{AB} \) are the relative clock offset and skew between the devices, respectively. Consequently, taking into account the datasheet of a typical crystal-quartz oscillator whose frequency varies up to 40 parts per million (ppm), the difference between clocks varies up to 40 \( \mu s \) per second [21], [22, Sec. 5.2]. In this paper, we acknowledge this problem and assume possible variations in the internal clocks.

Network Time Protocol (NTP) [23] is a method often used for synchronization. Despite the fact that NTP works fine in LANs, in cellular networks, the synchronization can be trickier. Since the ideal deployment of the proposed evaluation would be across cellular networks, we decided not to synchronize our devices using NTP. However, for the particular scenario presented, where the server is in the same LAN as the clients, using NTP is a feasible solution.
A. Timestamps

The server marks down a timestamp when the stream was received by the encoding application. Then, we encode and send out the information. As soon as the information is decoded, an acknowledgment is sent to the server, which marks down a second timestamp right after receiving the acknowledgment. Finally, the server computes the difference between the timestamps. We repeat the same procedure with the video service but without coding, to observe the extra latency of the system added by coding. Finally, we evaluate the round-trip time between the server and each client in order to subtract the time it takes to send the acknowledgment back to the server. We acknowledge the possibility that this method may not output exact results, since different loads may exist on the different link directions. Hence, we use a second method to contrast the results.

B. Snapshots

We run a clock in a display screen at the client. We point the recording camera to the screen, so that the streamed data is the running clock itself. Next, we play the stream in the same screen, and we take snapshots with an external camera to observe the time between the server and the client. We acknowledge the possibility that this method may not output exact results, since different loads may exist on the different link directions. Hence, we use a second method to contrast the results.

IV. Evaluation

In this section, we explain the setup used to perform the experiments and the parameters involved in the process.

A. Setup

The scenario consists of one server and multiple clients. Each coder runs in an Intel NUC6i5SYH. There is a streaming camera connected to the server that streams the recorded video. The server sends partial streams to each of the clients via Ethernet, and the clients use the WiFi access point (AP) to share their data using the wireless interface. In this particular case, the cellular phase will be emulated through Ethernet cables, while the cooperative phase will be wireless. The decoders are placed in the clients, which decode whenever they are able to. They send the decoded video to a VLC player to display the video in an LCD screen. A diagram of the setup is depicted in Fig. 3, and the hardware specifications are listed in Table II.

There is a USB video camera constantly streaming. This camera sends video chunks via UDP sockets, and the encoding application in the server encodes and sends them out as explained in Section II-C. We use VLC player [24] to stream and play the video. We use Real-Time Transport Protocol (RTP) as the transport protocol. This protocol is an IP-based protocol that runs over UDP [25], especially developed for multimedia streams. We use a transcoding protocol (H.265) on top of NC.

B. Configuration Parameters

We selected the testbed parameters such that the setup emulates a scenario close to reality. We use a power of 2 for the generation size, \( g \), and a Galois Field of \( 2^8 \), one of the most used values. We simulate errors during the cooperative phase by manually dropping packets at the receiver. The packet erasure rate (PER) is set to 0.1, the maximum value adopted in LTE [26, Sec. 7.2.3] and an often used value for WiFi [7]. To recover from this error, we adopt a minimum coding ratio of 120. Table III lists the most important parameter settings.

C. Results

In this section, we evaluate the latency overhead produced by our system under different cloud sizes. We compare only a cloud size of three cooperating clients and another cloud with
Latency evaluation for a three-client cloud

Latency evaluation for a four-client cloud

Fig. 4: Latency evaluation of the proposed NCC system for three-client and four-client clouds.

We first evaluate the latency with three clients. As can be seen in Fig. 4a, the coding latency moves in the range of hundreds of ms. Some peaks in the graph can be attributed to unrecoverable packets. During that period, the screen freezes and the video cannot be watched. Despite the peaks, RLNC redundancies allow the system to recover from non-bursty losses. In this case, the client that had to make use of redundancies will decode the information slightly later than the rest. If we combine this phenomenon with the caching queues existing in the camera and VLC players, we observe a permanent latency generated in the clients with the worst connectivity. The results can be seen in Fig. 4a, where Client 3 is the best, followed by Client 2, and lastly Client 1.

Lastly, we add a fourth client to the cloud and repeat the same evaluation. Fig. 4b plots 50 representative samples of the four-client configuration. In this configuration, we observe a higher likelihood of unrecoverable failures, plotted through the peaks in latency values. We observe a high correlation in the latency peaks, brought about connectivity errors and packet losses. Hence, we observe a clear error correlation in the clients in the MC. That is, the errors in one client highly impact the error likelihood in other clients. Regarding the recoverable errors, we observe the same behavior as in the three-clients case. The clients with a worse connection perceive the stream with a latency offset in comparison to the best one. The addition of the fourth client increases the average latency considerably, as reported in Table IV. It slightly destabilizes the system since both the standard deviation and the mean latency increase.

V. CONCLUSION

Current network infrastructures and protocols cannot bear the massive growth of devices and mobile data traffic. In particular, they inefficiently establish one unicast session per user, even if multiple users request the same data. NCC networks overcome this issue by mixing the advantages of NC and MC.

In this paper, we presented the latency evaluation of an NCC system by means of a testbed. We used two different methods to measure the latency, namely (i) the traditional timestamps, and (ii) a new method that does not require synchronicity between clocks. Evaluation results show a latency overhead of 200 ms and 400 ms for a cloud size of three and four clients, respectively. These results are acceptable for services such as video streaming, VR, and AR, that do not require immediate reaction from the user. We observed a high instability in terms of latency when the fourth client was added, which would lead to think that the more clients the more unstable the system will be. This would be solved by adjusting the number of RLNC redundancies so that no useless packets are sent through the network. We expect no significant latency increase if the number of RLNC redundancies match the channel losses. This, however, must be done periodically on the fly. We also observed a high correlation between losses and the different effects those losses have in terms of latency.

In the future, we plan to study the correlation between multiple MCs located next to each other. We will evaluate the possible interferences in the WiFi channel, and how this impacts the latency and the packet losses.

Table IV: Latency evaluation results for three and four clients in the cloud.

<table>
<thead>
<tr>
<th>Cloud Size $n = 3$</th>
<th>Cloud Size $n = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average (ms)</td>
</tr>
<tr>
<td>Client 1</td>
<td>290</td>
</tr>
<tr>
<td>Client 2</td>
<td>220</td>
</tr>
<tr>
<td>Client 3</td>
<td>194</td>
</tr>
<tr>
<td>Client 4</td>
<td>-</td>
</tr>
<tr>
<td>Average</td>
<td>235</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENT

This research has been supported by European Union’s H2020 research and innovation program under grant agreement H2020-MCSA-ITN-2016-SECRET 722424 [20] and the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany’s Excellence Strategy EXC 2050/1 Project ID 390696704 Cluster of Excellence Centre for Tactile Internet with Human-in-the-Loop (CeTI) of Technische Universität Dresden. The authors alone are responsible for the content of the paper.

REFERENCES