RC-NDN: Raptor Codes Enabled Named Data Networking

Carlos Anastasiades∗, Nikolaos Thomos†, Alexander Striffeler∗, Torsten Braun∗
∗Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland
Email: {anastasiades, striffeler, braun}@iam.unibe.ch
†School of Computer Science and Electronic Engineering, University of Essex, UK
Email: nthomos@essex.ac.uk

Abstract— Information-centric networking (ICN) has been proposed to cope with the drawbacks of the Internet Protocol, namely scalability and security. The majority of research efforts in ICN have focused on routing and caching in wired networks, while little attention has been paid to optimizing the communication and caching efficiency in wireless networks. In this work, we study the application of Raptor codes to Named Data Networking (NDN), which is a popular ICN architecture, in order to minimize the number of transmitted messages and accelerate content retrieval times. We propose RC-NDN, which is a NDN compatible Raptor codes architecture. In contrast to other coding-based NDN solutions that employ network codes, RC-NDN considers security architectures inherent to NDN. Moreover, different from existing network coding based solutions for NDN, RC-NDN does not require significant computational resources, which renders it appropriate for low cost networks. We evaluate RC-NDN in mobile scenarios with high mobility. Evaluations show that RC-NDN outperforms the original NDN significantly. RC-NDN is particularly efficient in dense environments, where retrieval times can be reduced by 83% and the number of Data transmissions by 84.5% compared to NDN.

Index Terms—Raptor codes, Information-centric networking, NDN, Opportunistic networks, Wireless communication

I. INTRODUCTION

Information-centric networking (ICN) is a new communication paradigm that aims at increasing security and efficiency of content delivery in modern communication networks. ICN introduces the concept of named content that can be stored or cached on any node. Routing is based on names instead of endpoint identifiers such that requests can be forwarded to and answered by the nearest (cached) copy. This feature is particularly attractive in mobile and opportunistic networks, where connectivity to nodes may be intermittent or even unpredictable. Hence, content requests can be directed to any content source in the vicinity of the nodes that request content.

In ICN, content can be retrieved from any neighbor node that holds the desired content and hence there is no need to maintain connectivity to a specific host. This enables ICN to fully exploit the inherent broadcast capability of wireless networks. However, ICN is not optimized for wireless broadcast or multi-path communication because the same copies of content are cached, retransmitted or routed along the communication paths. As a result, the limited resources on wireless medium and caches are not used efficiently because of redundant duplicate transmissions.

Recently, there have been some research efforts [1], [2], [3], [4] that aim at introducing advanced coding techniques such as network coding in ICN architectures to deal with inefficiencies of the ICN content delivery process. As it has been shown [2], [3], [4], network coding [5] assists ICN content delivery to decrease both the delivery times and the cache hit rates. The application of network coding in ICN is motivated by the benefits that network coding brings when employed in wireless communication systems [6], namely resilience to dynamics, improved throughput, decreased delivery times, etc. These benefits are mainly due to the fact that network codes can generate on-the-fly as many packets as are demanded. Also, it is beneficial that network coding flattens the importance of transmitted packets and simplifies packet scheduling. Although reported gains [2], [3], [4] are noticeable, these works deal only with static network environments. Furthermore, these works assume that all the network nodes are able and willing to perform operations with received packets, which may not be always the case due to computational constraints.

In this paper, we focus on networks characterized by significant mobility, e.g., moving cars or bicycles. We consider the Named Data Networking (NDN) architecture [7], which is a popular ICN architecture that follows a sequential request strategy, i.e., each content segment is requested in sequential order. Different from other works [1], [2], [3], [4] we apply Raptor codes [8], which like network codes also belong to the family of rateless codes. Raptor codes are applied only at the content sources and have very low computational cost. Thus, they do not introduce extra computational complexity in intermediate nodes. Furthermore, Raptor codes have linear encoding and decoding times and hence they are appealing for mobile networks that consist of low cost devices.

The biggest challenge with the introduction of Raptor codes to NDN is the design of a protocol that is compatible with the NDN request-response scheme, and supports pipelining to enable multiple concurrent requests in order to take advantage from the enhanced packet diversity that Raptor coding brings. The application of Raptor codes in NDN permits content to be requested in arbitrary order, which results in fewer duplicate transmissions than in original NDN, where multiple requesters can not take full advantage of each other because they request and store the same segments in their cache due to the sequential transmission policy. We show by simulations that Raptor
codes can significantly increase communication efficiency in dense wireless environments, such that requesters can send up to 92.7% fewer Interest messages and content sources send up to 84.5% fewer Data messages because requesters benefit from each other’s transmissions. This enables requesters to reduce content retrieval times by 83% compared to original NDN and to save resources on the wireless medium at the same time.

The remainder of this paper is structured as follows. In Section II, we describe the NDN architecture and present other NDN proposals that use advanced coding methods such as network coding. Additionally, limitations of existing designs are described. Section III presents our NDN design based on Raptor codes. Evaluation results are shown in Section IV. Finally, conclusions and future prospects are discussed in V.

II. RELATED WORK

In this section, we first present the basic components of NDN. Then, we outline some research initiatives on using advanced coding methods such as network coding in NDN. Finally, we discuss the limitations of existing approaches and motivate our work.

A. Basic NDN Concepts

NDN [7] is a novel network architecture that uses two types of messages for content delivery: Interests to request content and Data to deliver content. Typically in NDN, files consist of multiple segments that are included in Data messages. Then, in order to retrieve a file, network users need to express Interests in every segment of the original file. Please note that we use segments and chunks interchangeably throughout the paper. The most popular reference implementation of NDN is CCNx [9], which is open source. The core element of the implementation is the CCN Daemon (CCND), which performs message processing and forwarding decisions. Links from the CCND to applications or other hosts are called faces. The CCND contains three memory components:

1) The Forwarding Information Base (FIB) contains forwarding entries to direct Interests towards content sources. To avoid loops, Interests are never forwarded on the same face from where they were received.

2) The Pending Interest Table (PIT) stores unsatisfied forwarded Interests together with the face on which they were received. When Data is received in return, it is forwarded according to the face information found in the PIT. Existing PIT entries prevent forwarding of the same Interests.

3) The Content Store (CS) is used as cache in a NDN router. In the CS, received Data packets are temporarily stored.

Interests contain a field called Interest Lifetime that determines the maximum time that an Interest stays in the PIT, if no Data has been received in return. Since no Interests are forwarded for existing PIT entries, the Interest lifetime has direct impact on when Interest messages are re-expressed. Specifically, the file transfer proceeds as follows. First, an Interest is sent to request a particular segment, i.e., the Probing Interest. This is needed to check if a suitable content source is available. In case of a timeout, the Probing Interest can be retransmitted. The number of Interests that are concurrently transmitted is defined by the pipeline window. The latter increases through an additive increase multiplicative decrease (AIMD) mechanism for correctly received segments up to the pipeline size (maximum value). When a timeout occurs, the pipeline window size is reduced and the corresponding segment is requested again.

Prior to transmission, Data messages are included and scheduled for transmission in a queue. To avoid duplicate content transmissions, an additive random delay is considered during scheduling. If a node overhears the transmission of the same Data message from another node, it removes the scheduled Data from the content queue (duplicate suppression). Upon the reception of Data messages, they are cached in the CS for the time “freshnessSeconds”, which is specified in the header of the Data message.

In NDN, content names have a hierarchical structure composed of multiple (arbitrary) name components, a file name, a version number that labels different versions of the same content and a segment number to mark each segment individually. The name structure looks as follows:

/\texttt{c}_0/\cdots/\texttt{c}_n/\texttt{fname/vers/s}_n

where \texttt{c}_0, \cdots, \texttt{c}_n denote arbitrary name components, \texttt{fname} is the file name, \texttt{vers} stands for the version and \texttt{s}_n is the \texttt{n}th segment number.

B. Network Coding in Information-Centric Networks

The potential of network coding in Information-centric networks has been first discussed in [1]. Random linear network coding has been used to encode multiple packets (chunks) together. Hence, instead of sending the original (uncombined) packets, senders transmit network coded packets over multiple paths. Network coding decreases the probability to send duplicates of the same packet and increases packet diversity in the network. Therefore, received packets have higher probability to be useful for decoding. Network coding requires additional header fields in both Interest and Data messages: Interest messages contain a flag indicating whether the response should be network coded and Data messages require an additional header with the coefficients used for encoding.

Motivated by [1], CodingCache [2], a caching extension with network coding, has been proposed. Simulations show that network coding can increase cache hit rate due to cache diversity. This is due to the re-encoding process (network coding), which is performed in intermediate nodes to improve packet diversity. Furthermore, Interests are forwarded uniformly at random to maximize the overall cache hit rate. An analytical study on network coding in NDN [3] proposes offline solutions that increase network efficiency by dynamically exploiting network-coded caching and multicasting. The solutions are examined in a butterfly network and the evaluations show that random linear coded caching and multicasting is sufficient for achieving minimum cost caching-aided multicast.
More recently, the design of a controller-based framework that employs linear network coding for efficient cache management and content routing has been proposed [4].

C. Limitations of Existing Approaches

Although existing works [1], [2], [3], [4] seem to be very promising, they do not consider the implications of the proposed modifications on (and limitations of) Information-centric networks. All the approaches use additional headers for network coding, and process the messages similar to regular network traffic. These works disregard that the foundation of Information-centric networks are names, and thus, message processing and matching should be based on content names and not based on flags in messages. Since Interests need to match content names, an identifier in the name is required to distinguish the content type, as encoded and non encoded content have different structures.

The matching process in NDN is deterministic: if the same Interest is retransmitted, the same segment will be retrieved. However, existing network coding approaches assume that during the matching process a random combination of chunks is encoded and returned. If matching is turned into a random process at the CCND, retrieving missing pieces of information becomes more complex. In such a case, Interest messages should include additional information about already received chunks to avoid the retrieval of duplicate content. Exclusion filters have been proposed [10], but they have disadvantages. Specifically, the size of Interest messages would increase with the number of excluded chunks and this would introduce additional delays in the transmission. Furthermore, processing of Interest messages would become more computationally demanding, especially when long exclusion filters are considered during matching operations. Finally, multiple Interests with the same request prefix would not be possible when exclusion filters are used because of the following reasons. First, the PIT would prevent forwarding of multiple Interests with the same prefix on the same face. Exclusion filters are not considered for forwarding, but only in the matching process. Second, if multiple Interests with exclusion filters would be transmitted, only one Data message could satisfy all Interests if it is not included in the exclusion filters. Therefore, Interests with exclusion filters need to follow a stop-and-wait strategy, i.e., Interest transmission after Data reception, and cannot use pipelining.

Existing works [1], [2], [3], [4] do not consider security aspects of NDN. Since the content publisher signs every segment, re-encoding by means of network coding is equivalent to generating new content with a new signature. Thus, re-encoding can not be performed transparently at intermediate nodes without requesters noticing it. If the generated network coded packets are not signed, malicious nodes could interfere with the communication by introducing single bogus messages. If nodes that apply network coding would sign messages, transitive trust models would be required.

III. RAPTOR CODING FOR NDN

In this section, we present our proposed Raptor coding based NDN protocol. We first give a brief overview of the employed Raptor codes [8] and then discuss our RC-NDN architecture emphasizing on the required data structures and names. We also provide a sample request procedure.

A. Raptor Codes

Raptor codes belong to the family of fountain codes [11]. They are particularly appealing for real-time data delivery both in wired and wireless networks. Unlike other fountain codes, Raptor codes have linear encoding and decoding times, hence, they are appropriate for real-time communication. Raptor codes are endowed with the rateless property that permits the generation of a potentially unlimited number of packets from a given set of source packets K. Raptor encoding has two phases. First, packets are encoded by a pre-coder, e.g., a LDPC [12] encoder, and then the resulting (pre-coded) packets are fed into a LT encoder that combines them by means of XOR coding to generate the Raptor encoded packets. The number (degree) of combined packets depends on a carefully designed degree distribution function. Raptor encoded packets are augmented with a header that can be the seed of a pseudorandom generator used for deciding which packets are combined to generate an encoded packet. This header is required, as Raptor codes have an implicit coding structure, i.e., neither the number of encoded packets is known a priori nor the packets that have been encoded together. Raptor codes perform very close to Reed-Solomon codes and are characterized by a very small communication overhead that is typically in the range [3%, 5%]. Hence, \(K \cdot (1+e)\) packets should have been received for successful decoding. Obviously, Raptor codes have only probabilistic guarantees of decoding, i.e., when the number of received packets \(N\) exceeds \(K\), Raptor codes can be decoded with a probability that increases with \(N\). When these codes are used in networks with diversity, e.g., overlay networks, the communication systems take advantage of enhanced packet diversity to cope with loss without requiring ARQ messages. Although, Raptor codes are traditionally applied end-to-end, they can be combined with network codes [13] to further enhance the resilience of communication systems.

B. Packet Processing

The Raptor coding process can be included either in the CCND or in an application module above the CCND. If it would be included in the CCND, encoding, message collection, and decoding should also be done at the CCND. This approach would have the advantage that Raptor coding is transparent to the application; however, it is associated with two main drawbacks. First, an application does not know a priori how many Interests have been transmitted and received. Requesting new messages should, therefore, be the CCND’s responsibility. In order to detect when content retrieval has been completed, the application should make regular checks. This means that the CCND would need to keep track of complete and partial content and provide it to applications.
upon request. However, these checks at the CCND increase the computational complexity significantly. In addition, received, decoded, and re-encoded data would need to be stored in the cache, which is problematic because it is only a temporary storage. Hence, data may be deleted before a sufficient number of packets for decoding has been received, which depends on cache replacement strategies. Since only applications store content in persistent file systems, they need to request and receive content in the first place.

The second option is to include Raptor coding as application layer module above CCND. Requesters would store all encoded packets temporarily in the content store and persistently at the application layer. A disadvantage of this approach is that only requesters and content sources can perform encoding and decoding operations, while intermediate nodes only store received packets in and relay them from the cache.

As mentioned in Section II-C, re-encoding requires new signatures, which may raise trust issues. Therefore, we decided to follow the second approach, i.e., to introduce Raptor codes as an application layer module. We assume that encoded content has the same name prefix, which means that the content is created by the same publisher. We consider only coding of content segments from the same file or stream and not coding between content published by different publishers.

C. Data Structures

In the proposed RC-NCN architecture, we do not modify the original NDN Interest and Data messages. The information that is needed to request Raptor coded packets is included in the Interest name as explained in the next subsection. Original Data messages are serialized and encapsulated as payload in a new message structure called Raptor Data message. The structure of Raptor Data messages is similar to original Data messages and contains the same headers and signatures as with regular NDN. Additionally, we append a Raptor coding header (RCH) to enable decoding at the requesters side. The RCH conveys the following information:

1) The seed used for generating the Raptor coded packets.
2) The number of generated Raptor coded packets \( N \).
3) The number of original source packets \( K \), i.e., segments.

D. Naming

Since message processing in NDN is based on names, identifiers of encoded messages should be included in the content name. To differentiate between regular traffic and Raptor encoded packets, we have included an additional marker \( \text{rc} \) after the file name. The proposed name structure is as follows:

\[
/\text{name}/_{rc}/\text{version}/_{enc}/e_0/\cdots/e_n/\text{name}/ \]

where \( e_n \) stands for the \( n \)th combination of packets, i.e., encodedID. To avoid the reception of packet duplicates, the requester specifies the encodedID in the Interest name. The first combination \( e_0 \) equals the seed of the encoding and \( e_n \) is increased for every additional packet combination.

Without specifying the encodedID \( e_n \), multiple parallel Interest transmissions (pipelining), are not possible as explained in Subsection II-C.

E. Request Procedure

The Data request procedure works as follows. The requester first transmits a general Interest in \(<\text{content_name}>/\text{rc} \) without encodedID. This Interest requests the lowest available encodedID of the content name. The requester receives for example a Data message with the name \(<\text{content_name}>/\text{rc}/\text{version}/25034 \). By looking into the RCH, the requester discovers both the seed that was used in the encoding process and the number of encoded packets. With this information, the requester can directly demand multiple encodedIDs simultaneously by increasing the encodedID in the requests. If there is a timeout, i.e., no reply to an Interest due to collisions or content unavailability, the encodedID is increased and the next encodedID is requested. When the maximum encodedID is reached, which is equal to the sum of seed and overall number of encoded packets, the next encodedIDs to be requested start over from the seed with combinations that have not yet been received.

IV. Evaluation

In this section, we evaluate the performance of the proposed RC-NDN architecture and compare it to original NDN. We use our NDN simulation framework, which has been implemented in OMNeT++ [14] in order to facilitate network setup and enable high scalability with respect to network size. The framework provides a complete NDN implementation including all memory components such as the content store (CS), pending interest table (PIT), forwarding interest base (FIB) as well as content queues with the same delays and flags as in CCNx. Therefore, the framework allows us to process and store information like CCNx except for signature calculations and verifications, which are omitted for simplicity.

A. Simulation Scenario and Parameters

The evaluation parameters are listed in Table I.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface</td>
<td>1x 802.11g</td>
</tr>
<tr>
<td>data rate</td>
<td>2Mbps</td>
</tr>
<tr>
<td>error rate</td>
<td>Nist error rate model</td>
</tr>
<tr>
<td>transmitter power</td>
<td>2.0mW</td>
</tr>
<tr>
<td>signal attenuation</td>
<td>free space path loss</td>
</tr>
<tr>
<td>playground size (square)</td>
<td>300m, 1000m, 2000m</td>
</tr>
<tr>
<td>mobility</td>
<td>Gauss-Markov, ( \alpha = 0.9 ) speed: 10m/s</td>
</tr>
<tr>
<td># requesters</td>
<td>10, 30, 50, 70, 100</td>
</tr>
<tr>
<td>segment size</td>
<td>4096 Bytes</td>
</tr>
<tr>
<td>content size</td>
<td>1024 Bytes</td>
</tr>
<tr>
<td>content lifetime CL</td>
<td>60s</td>
</tr>
<tr>
<td>pipeline size</td>
<td>16</td>
</tr>
</tbody>
</table>

We use a Gauss-Markov mobility model with a node speed of 10 m/s. Nodes can overhear transmissions up to a distance of 240m. Each mobile node seeks for content provided by a mobile content source. Upon the reception of the first request, the content source encodes the content by means of Raptor coding and transmits Raptor encoded messages to all
requesters that are in one-hop distance. We use Raptor codes with 3GPP degree distribution. Only content sources perform encoding, while requesters do not perform any re-encoding operations. As in CCNx, we delay content transmissions randomly with a delay in the interval [50ms, 150ms] to enable duplicate suppression. All requesters randomly start the request phase within the first [1,100] seconds of the simulations by expressing Interests in the same content name, i.e., they ask for the same data. In our original NDN implementation, Interests that time out are retransmitted twice. If the Interests are not satisfied by the second retransmission, the next Interest is transmitted after a probing interval of 4 seconds. In the RC-NDN, users request the next encodedID in case of a timeout. All reported results are averaged over 100 simulations.

B. Content Retrieval Times

We first investigate the cumulative retrieval times for both RC-NDN and original NDN. We define the cumulative retrieval time as the time at which a specific number of nodes has completed content retrieval.

Fig. 1 shows the cumulative retrieval times of 100 nodes that request content using RC-NDN and original NDN in square playgrounds of various sizes. The reported playground values correspond to the side that is measured in meters. The x-axis shows the time in seconds and the y-axis the cumulative number of nodes that have completed the content retrieval.

In Fig. 1, we can see that RC-NDN is considerably faster than NDN for all the examined playground sizes. For example, in a playground of 500m length, RC-NDN is 5.8 times faster than NDN, i.e., 990s instead of 5792.3s, while in a playground of 4000m, it is still 4.7 times faster, i.e., 1380.6s instead of 6524.5s. There are two main factors that limit NDN’s retrieval time in broadcast environments. First, the PIT table prevents the transmission of same Interests for the entire duration of an Interest lifetime. This means that a node in vicinity of a content source may not be able to transmit Interests when it receives the same Interests from another node before sending its own. As a consequence, due to the limited wireless transmission range, a node may overhear unanswered Interests from another requester preventing the transmission of its own Interests although a content source would be reachable. Differently from original NDN, in RC-NDN Interest transmissions are less likely to be blocked by the PIT since requesters do not request content sequentially. Second, in original NDN multiple nodes often simultaneously request the same segment. However, due to the duplicate suppression procedure explained in Section II-A only one segment transmission is performed, which limits the performance of NDN. In RC-NDN, nodes usually request different segments. Thus, multiple transmissions can be performed in parallel such that requesters can profit more from each other’s request. In Fig. 1, we further observe that retrieval times of RC-NDN increase with increased playground sizes, i.e., there is a time increase of 39% from a playground with side length 500m to a playground with side length 4000m. A similar behavior is noticed for the NDN protocol. However, in the NDN case when the requester density is too high, e.g., on the 500m-playground, the retrieval requires more time than on sparser playgrounds such as 1000m or 2000m playgrounds because of a higher collision probability and more duplicate transmissions when requesting and caching the same segments.

In Figs. 2a and 2b the cumulative retrieval times of RC-NDN and original NDN are presented when only a fraction of nodes request content in playgrounds of sizes 500m and 4000m with 100 nodes. In Figs. 2a and 2b the cumulative retrieval times of RC-NDN and original NDN are presented when only a fraction of nodes request content in playgrounds of sizes 500m and 4000m. Fig. 2a shows that when the node density is high, RC-NDN enables all requesters to retrieve the content approximately at the same time. This occurs independently of the effective number of requesters. In contrast, the NDN retrieval time increases with increasing number of requesters. As can be seen, the curves are shifted on the x-axis. This means that the same number of requesters needs more time to retrieve content when there are more requesters, i.e., higher total number, in a network. From Fig. 2b, we can observe that for low node density values (large playgrounds), the number of requesters has a greater impact on the retrieval time. The more requesters there are, the faster is the content retrieval. Specifically with RC-NDN, content retrieval in networks with 100 requesters...
is 4.6 times faster than in networks with 10 requesters, i.e., 1380.6s instead of 6407.9s. In NDN, the retrieval times for networks with 100 requesters are 1.8 times faster than networks with 10 requesters, i.e., 6524.5s instead of 11828.9s. Even in low density networks, RC-NDN performs better than NDN in terms of content retrieval time, however, the relative improvement is lower than in high density networks.

C. Transmitted Messages

In this section, we evaluate RC-NDN and NDN in terms of transmitted messages. The more messages are sent, the more energy is consumed for message processing and wireless transmission. In Fig. 3, we illustrate the number of transmitted Interests in different playground sizes when 100 requesters are considered. We note that in dense environments RC-NDN results in a considerably lower number of transmitted Interests. For example, in a playground size of 500m, each requester in NDN needs to transmit 1303 requests (average value) to complete content retrieval, while using RC-NDN only 95 Interests are needed on average. RC-NDN requesters can send fewer Interests because, due to increased request diversity, they can benefit more from each other’s transmissions. Hence, we conclude that RC-NDN drastically reduces the required number of requests in dense environments. In bigger playgrounds, i.e., 4000m, with RC-NDN the average number of Interests is still 17.8% lower than when NDN is used, i.e., 1293 Interests instead of 1572, but the variation of RC-NDN is higher. The reason for this high variation is our selected Interest transmission strategy for RC-NDN, which does not adapt the pipeline window and continues to send the next Interests in case of timeouts. If no content source is in the vicinity of a node, multiple unnecessary Interests are transmitted, which are not answered.

We have also performed evaluations with a lower number of requesters, but the graphs are omitted due to space limitations. Evaluations show that in case of 10 requesters in a 500m playground, requesters with RC-NDN send 38% fewer Interests than NDN. The drop of transmitted Interests for RC-NDN reaches 92.7% when 100 requesters are assumed. In larger playgrounds (i.e., 4000m) and 10 requesters RC-NDN sends 2.7 times more Interests on average, while for 100 requesters RC-NDN results in 17.3% fewer Interests than NDN.

Interest messages are in general smaller in size compared to Data messages that contain content to be exchanged. Hence, we next study Data message transmissions in detail. In Fig. 4, we depict the number of transmitted Data messages by requesters and content sources for various playground sizes that contain 100 requesters. Evaluations show that the number of Data messages transmitted by each requester using RC-NDN increases slightly with the playground size. This is attributed to a lower requester density, since requesters can also provide content to other requesters. With NDN, the behavior is quite different. Since requesters using original NDN demand and store content sequentially, broadcast requests may result in a significantly higher number of duplicate transmissions in dense environments. This phenomenon is less pronounced in sparse networks (large playgrounds).

In all scenarios, RC-NDN requesters need to transmit fewer Data messages than NDN requesters. Specifically, in a playground size of 500m, RC-NDN requires on average 95.8% fewer Data messages than NDN, i.e., 162 instead of 3854. In a playground of 4000m, RC-NDN still needs on average 36% fewer Data packets than NDN, i.e., 791 instead of 1236. Due to the increased content transmission diversity, content sources can send fewer Data messages as well. For playgrounds of 500m, a content source using RC-NDN transmits 84.5% fewer Data messages, i.e., 1531 instead of 9857, and for a playground of 4000m approximately 26.7% fewer Data messages, i.e., 6246 instead of 4577.

In Figs. 5a and 5b, we present the number of Data messages transmitted by requesters and content source in playgrounds of 500m and 4000m for a varying number of requesters. As Fig. 5a shows, in dense environments the number of transmitted Data messages by content source and requesters using NDN increases with increasing number of requesters, while it increases only slightly with RC-NDN. For 10 requesters, content source and requesters transmit on average 59.9% and 95.1% fewer Data messages with RC-NDN compared to NDN. In case of 100 requesters, content source and requesters transmit even 84.5% and 95.8% fewer Data messages. These results show that RC-NDN can significantly reduce the number of Data transmissions in dense environments, and by that, reduce the number of (unnecessary) duplicate transmissions.

In playgrounds of 4000m with 10 requesters, a content source in RC-NDN needs to send slightly more Data packets, i.e., on average 4.9%, due to a lower requester density and requesters not overhearing each other. As mentioned in Section
III-A, RC-NDN requesters need to receive $\epsilon \times K$ additional packets to successfully decode content. However, if the number of requesters increases to 30 requesters or more, a content source sends up to 26.8% (for 100 requesters) fewer Data messages, even in dense environments.

V. CONCLUSIONS

In this paper, we have studied the application of Raptor codes to Information-centric networks to decrease content retrieval times in wireless networks characterized by high mobility. We have based our implementation on the popular NDN protocol. From the evaluation, we have seen that the introduction of Raptor codes in NDN protocols offers large gains compared to original NDN in terms of content retrieval times and the number of transmitted Interest and Data messages. Larger gains are noticed in dense networks even if the number of requesters is very small. Interestingly, the performance of RC-NDN increases with the number of requesters. The superior performance of RC-NDN over NDN is attributed to the inherent inefficiency of NDN to deal with a high number of requesters. In such cases, the PIT of NDN blocks multiple Interests in the same segments. Therefore, due to limited wireless transmission ranges, a node that overhears (unanswered) Interests from another requester may not send its own requests although a content source would be reachable. Additionally, due to the sequential requesting policy in NDN, many redundant Data messages are transmitted and cached. This leads to an increased number of collisions, duplicate transmissions and eventually to lower transmission speed. The improved performance of RC-NDN renders it particularly appropriate for dense urban environments, such as train stations or sports stadiums, where multiple requesters can benefit from each other when requesting popular content.

Although RC-NDN offers noticeable gains in sparser environments, these are smaller than in dense networks. In case of only a few requesters, RC-NDN may even perform worse in terms of message transmissions because requesters may not overhear each others’ transmissions and need to receive slightly more messages for decoding. In addition, requesters may transmit considerably more Interests than with NDN. This is due to our Interest transmission strategy that does not adapt the pipeline window when timeouts occur, but proceeds with the next encoded message. We argue that exclusion filters may increase the size of Interest messages and avoid pipelining. However, there is a trade-off between Interest and pipeline window. Hence, in case of multiple timeouts in a row it may be better to adapt the transmission strategy to send one large Interest with exclusion filters than many small Interests. As soon as content is received, pipelining could contine as explained in this paper.

We close this section with the following remark. Although we have considered single-hop communication and content is Raptor encoded only at the content source (recall that network coding at requesters or intermediate nodes is not permitted to avoid new signatures and establish transitive trust models), we expect that the introduction of network coding capabilities to requesters and intermediate nodes could further enhance the performance, especially in multi-hop networks.

REFERENCES