MICN: a Network Coding Protocol for ICN with Multiple Distinct Interests per Generation

Hirah Malik, Cedric Adjih, Claudio Weidmann, Michel Kieffer

MICN: A NETWORK CODING PROTOCOL FOR ICN WITH MULTIPLE DISTINCT INTERESTS PER GENERATION *

Hirah Malik
Inria, Université Paris-Saclay, L2S
1 Rue Honoré d'Estienne d'Orves,
91120 Palaiseau,France
hirah.malik@inria.fr

Cédric Adjih
Inria
1 Rue Honoré d'Estienne d'Orves,
91120 Palaiseau,France
cedric.adjih@inria.fr

Claudio Weidmann
ETIS UMR8051, ENSEA,
CY University, CNRS,
95000 Cergy, France
claudio.weidmann@ensea.fr

Michel Kieffer
Université Paris-Saclay,
CNRS, CentraleSupélec, L2S,
91192 Gif-sur-Yvette, France
michel.kieffer@l2s.centralesupelec.fr

ABSTRACT

Information-Centric Networking (ICN) is based on a recently proposed family of protocols in which contents are identified through names. ICN decouples the content to be retrieved from the specification of its location. It naturally supports the use of multiple paths; however, with multiple consumers and producers, coordination among the nodes is required to efficiently use the network resources. Network coding (NC) is a promising tool to address this issue. The challenge when using NC with ICN is to be able to get independent coded content in response to multiple parallel interests (*i.e.*, requests) by one or several consumers. In this work, we propose a novel construction called MILIC (Multiple Interests for Linearly Independent Contents) that imposes constraints on how the replies to clients are network coded, intending to get linearly independent contents in response to multiple interests. Several protocol variants, called MICN (MILIC-ICN), built on top of NDN (Named Data Networking), are proposed to integrate these interest constraints and NC of data packets. Numerical analysis and simulations illustrate that MILIC performs well and that the MICN protocols reach close to optimal throughputs on some scenarios. MICN protocols compare favorably to existing protocols and show significant benefits when considering the total number of transmitted packets in the network, also in the case of high link loss rate.

1 Introduction

Content distribution has become the primary task for today's Internet. According to CISCO's forecast, video traffic will be accounting for 79 percent of total mobile data traffic by 2022 [1]. The communication network's traditional paradigm has some drawbacks, especially when dealing with large-scale content distribution because of the point-to-point nature of communication and location dependence. The consumers, however, care about the content itself and not about its origin.

Information-Centric Networking (ICN) has recently been proposed as an alternative to the traditional point-to-point communication to make content the center of the communication network [2]. The ICN principle is based on naming contents instead of their locations. Thus, it removes the need to establish a connection between endpoints and allows caching throughout the network. Content Centric Networking (CCN) [3] and Named Data Networking (NDN) [4, 5] are two examples of ICN architectures.

*Accepted Manuscript, final version in *Computer Networks* journal, at DOI: 10.1016/j.comnet.2021.107816

The basic NDN framework is a pull-based mechanism where the clients send interest packets that contain the name of the requested content. These interest packets are routed based on their names. A node holding a copy of the requested content replies to the interest with the content in a data packet.

Devices nowadays come with multiple network interfaces that can be used to retrieve content, *e.g.*, WiFi, 4G/LTE. Traditional networking requires establishing a session among endpoints. Using multiple paths is possible in IP, *e.g.,* with MPTCP [6], but it requires associating multiple paths to multiple addresses. In contrast, NDN inherently enables the use of multiple interfaces. Nevertheless, in a dynamic network with multiple clients, some coordination is required to take advantage of multiple paths. Montpetit *et al*. introduced network coding (NC) over ICN [7] as an alternate to the coordination approach.

NC is a communication paradigm which, unlike traditional networking, allows the nodes in the network to perform coding operations on the packets [8]. Random linear network coding (RLNC) is one of the simplest ways to combine packets in the network by performing linear combinations with randomly selected coefficients [9, 10]. Decoding is performed by solving a linear system of equations once enough linearly independent combinations/packets are received. NC helps to exploit the network capacity, minimize delays and may help recover from link failures. Traditional routers that could only forward or replicate the packets are replaced by coding routers that can mix packets.

In this work, we aim to integrate more efficiently NC within ICN, focusing specifically on NDN. For that purpose, we introduce an index in each interest that imposes constraints on the coded segments that can serve as a reply to that interest. This construction of interests is called MILIC (Multiple interests for Linearly Independent Content). The MILIC constraints ensure retrieval of linearly independent content with each interest. Several protocol variants, called MICN (MILIC-ICN), are then built using the MILIC construction. MICN protocols allow parallel processing of multiple interests sent by nodes and ensure linearly independent content with each of the multiple interests. The MILIC construction is confirmed to perform well by numerical analysis and simulations. On considered scenarios, the MICN protocols manage to approach the optimal throughput. MICN compare favorably to existing protocols and show significant benefits in terms of the total number of transmitted packets in the network, and in the case of high link loss rate.

Section 2 summarizes some related work. Section 3 details the construction of the MILIC interests. In Section 4, we detail the MICN protocol that uses MILIC construction to integrate NC over NDN as well as some optimizations to further improve the performance of the protocol. Section 5 presents the simulation setup and results. Section 6 concludes this paper and introduces future work.

2 Related Work

2.1 NDN

In this section, we briefly explain the basic concepts of the NDN architecture [4, 5]. Communication in NDN is *consumer-driven,* with two basic types of communication packets: *interest* and *data* packets. Consider an NDN network consisting of a set $\mathcal{N}$ of nodes, that can be *sources* that generate content, intermediate nodes or *caching routers*, or *client*s that request content. A node can have any of these roles at a given time. Each node $r \in \mathcal{N}$ is connected in the network through a set of faces $\mathcal{F}_r$. The term *face* is a generalization of the interface that corresponds to various communication links.

The clients request the network to find content by sending an NDN interest packet that carries the name of the requested content. The interest is forwarded in the network until it reaches a node holding a copy of the content with the requested name. The content is then sent back in a data packet. Both interest and data packets carry the name of the content but there is no information regarding the client or source.

Each NDN node has a *Pending Interest Table* (PIT), a *Forwarding Information Base* (FIB), and a *Content Store* (CS) for the transport of the named content in the network [3]. The PIT keeps a record of pending interests forwarded by the node that are not satisfied yet. Along with the interest, the PIT stores the face where each interest arrives (in-faces) and the faces to which it was forwarded (out-faces) to record the reverse link for the data packet. The FIB stores routing information used to forward interest packets toward potential sources of matching content; it can be populated by self-learning or by a routing protocol. Routing and forwarding strategies to efficiently perform the named-based routing are presented in [11] and [12]. Finally, the CS is a cache memory. An intermediate node can decide to cache the content that it forwards downstream for replying to future interests. Consequently, content is stored in source nodes and in caching routers [4].

Each interest packet brings back one data packet if a copy of the requested content is found. If the content object is large, it may be partitioned into smaller segments to fit into data packets. In a classical NDN, the client

requests a content segment by sending the name prefix with the segment identifier [13]. For example, the interest `<content-name>/<i>` is requesting the i^{th} segment of a content. Each interest also carries a random identifier, *nonce*, that helps to prevent interest forwarding loops [4]. Interests are forwarded using the information in the FIB.

A node that receives interest for a segment verifies that no similar interest is pending in its PIT. Having a pending interest means that the requested content is not in the CS, so the node updates the pending entry in the PIT by adding the receiving face of the new interest. If there is no pending entry with the same name, the node then checks its CS for a copy of the requested content. If there is a *cache hit, i.e.*, the requested content is available in the cache, the node replies to the interest with a data packet. In the case of the unavailability of requested content or *cache miss*, a new PIT entry is created, and the interest is forwarded to the available faces in the FIB. Once the content is received from upstream, it is routed back to the requesting node using the information in the PIT. The node also decides whether the content should be cached locally [4].

2.2 Network Coding and Interest-Based ICN

NC and ICN both inherently tend to address the content delivery and focus on improving content distribution over the network. NC and ICN can work jointly to exploit network capacity better (by exploiting caching, multi-path delivery, *etc*.). The idea of applying NC over ICN was first introduced by Montpetit *et al.* to take advantage of ICN and NC's inherent features to improve content delivery [7]. They described NC implementation over a Pub/Sub ICN architecture [14] and an interest-based architecture (CCN) with NC3N. There is currently ongoing work in standardization on combining NC and ICN [15].

We focus on interest-based architectures. In a NC scenario, the original content is partitioned into smaller groups of segments, called *generations.* NC is only allowed among the segments of the same generation to reduce the decoding complexity. With RLNC, segments from a given generation may be linearly combined within a source or at any intermediate node in the network. The linear combinations are performed in some Galois field $\mathbb{F}_q$ to get *coded* segments. In what follows, $\mathbb{F}_q$ is a Galois field of q elements and $\mathbb{F}_q^*= \mathbb{F}_q \setminus \{0\}$. The coefficients in $\mathbb{F}_q$ of the linear combination form the *encoding vector* [16] of each coded segment. In the NDN context, the source nodes and caching routers may store original and coded segments.

The client nodes send interests requesting coded segments instead of a specific segment. The name carried by interest packets is adapted to indicate that coded segments are expected (*e.g.*, by setting a flag that indicates the retrieval of coded segments [7]). Requesting coded content allows the intermediate or source nodes to send different coded segments generated by combining the original content segments in their cache, instead of one particular segment.

The pull-based request, response mechanism of NDN allows one interest to bring back one content segment. A client node has to send at least n interests to be able to decode a generation of n segments. Based on NDN interest processing, it is challenging to ensure retrieval of *innovative* (linearly independent) content with each interest, required to keep minimal decoding delay and network load.

In the coded NDN schemes proposed in [17, 18, 19], encoding vectors of all the received coded segments are sent in the interests. The encoding vectors help the nodes to either generate a coded segment that will be innovative for the requesting node or forward the interest to their next-hop neighbors. Nevertheless, this approach introduces an overhead in the interest packets that increases with each coded segment the requesting node receives. The size of the overhead is limited by keeping the generation size small. This approach also introduces a delay as the client node waits to receive the replies for previously sent interests to arrive before it can issue interests for more coded segments to ensure retrieval of linearly independent content with each interest.

Zhang *et al*. compared the approach of sending all received coefficients (precise matching) to rank-based matching, *i.e.*, sending only the client node's rank. They observed that rank-based matching achieves slightly lower performance but has much lower computation and communication overhead [20].

Liu *et al*. [21] introduced an interest coding and forwarding strategy that allows splitting and joining interests for the same content and generation at intermediate nodes. The interests request a subset of segments by indicating the number of required coded segments to get a decodable generation. This scheme implements a one interest-multiple replies strategy, which is not compliant with the NDN principle of one interest-one reply.

NetCodCCN [22] addresses the shortcomings of previous approaches by sending undifferentiated interests for coded segments of a generation. The client node implicitly states that it requires another coded combination by sending additional interests for coded segments. The intermediate nodes that have previously sent coded segments keep track of the number of coded segments forwarded on each face and the rank of the set of linear combinations in their CS. The node only replies to an interest if the rank of its CS is larger than the number of coded segments it has sent on a particular face. NetCodCCN also supports the transmission and handling of multiple interests at one time (pipelining),

to allow nodes to request content more efficiently. With pipelining, a burst of interests is sent first by a client. Each time content is received, a new interest is sent. A drawback of this approach is unnecessary data traffic that flows in the network after the clients have received enough content to decode a generation.

Liu *et al*. [23] added a parameter r indicating the number of desired coded segments in the interest requesting more than one coded segment, which is again contrary to the NDN principle of one interest-one reply. Matsuzono *et al*. [24] proposed L4C2, a low-loss, low-latency, NC enabled video streaming protocol over CCN. In L4C2, nodes request network coded packets in case of data packet losses only. Bilal *et al*. [25] proposed an algebraic framework of NC over NDN.

3 Proposed approach for Network-Coded NDN

In the classical NDN framework, there is a one-to-one mapping between the requested content and interests. For a content split in n segments, a client sends n interests, each requesting a specific segment by stating its unique name and segment number. The replies to these interests are the requested segments.

With NC, coded segments are stored at various places of the network. As observed, *e.g.*, in [7], many different coded segments could serve as a reply to a given interest. The main challenge when sending interests for coded segments is to ensure that linearly independent segments are sent as replies. This problem is challenging since any intermediate node that has cached coded segments from a generation can generate one or more coded segments and hence can reply to several interests from the same client, but linear independence among the replies is not ensured.

As mentioned, in some previous work [17, 18, 19], only one outstanding interest is allowed at the expense of delay. Alternately, [23] reduce delay and overhead by requesting multiple coded segments in one interest. Saltarin *et al.* [22, 26] overcome the delay problem by pipelining multiple identical interests and constraining the number of coded segments sent on a face, at intermediate nodes.

In this work, we propose a protocol called MICN, that supplements NDN with NC. In MICN, the client nodes implicitly indicate the required coded segments by pipelining multiple distinct interests. The distinct interests allow parallel processing of the multiple interests and ensure that replies to each of the pipelined interests are not redundant.

3.1 MICN Architecture

The architecture of MICN is based on NDN [4]. It incorporates common adaptations of NC to NDN (as described in Section 2.2), including dividing content into generations and allowing in-network operations on the content from one generation. Fig. 1 presents the basic elements of MICN. For the protocol messages: interests are sent to retrieve coded segments, and replies to these interests are coded segments (*i.e.,* linear combinations of the original segments) sent in data packets. For the information bases: the nodes have a PIT, a CS, and a FIB, modified to receive, process, and forward coded segments. Original or coded segments may be stored at the source and intermediate nodes. Each generation is retrieved independently of others.

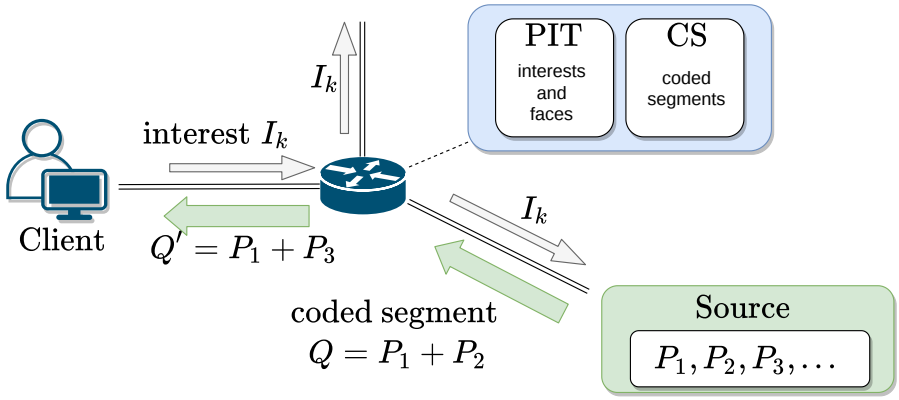


Figure 1: Basic elements of MICN

One central design goal of MICN is to achieve high throughput using multiple available paths to the sources. To achieve this, MICN maximizes the propagation of interests. The main MICN semantics are explained below.

- **Forwarding strategy:** MICN uses a *multicast forwarding strategy*: at each node, when necessary, interests are forwarded on all available faces.
- **Pipelining:** Interests are pipelined to allow parallel retrieval of content and hence increased throughput.
- **Interest processing:** Interests of different clients (similar interests, different nonces) are not suppressed: this effectively lets the interests of each client reach the sources from multiple available paths.

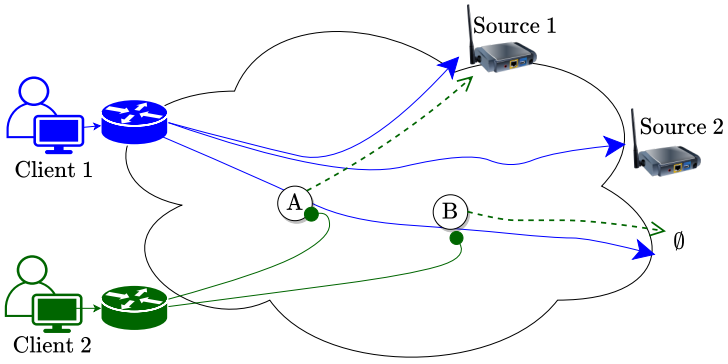


Figure 2: Impact of interest suppression

MICN, unlike NDN, chooses not to suppress the forwarding of similar interests coming from different clients and treats each interest with a different nonce as a new interest. The reason for not merging is illustrated in Fig. 2 where interest I_k from client 1 is propagated in the network first. With the plain NDN semantics, it stops the propagation of similar interest from client 2 since the same content is expected from the client 1 interests. As a result, client 2 will be unable to take advantage of the multiple paths and get content from only one path (from node A in Fig. 2).

In order to achieve high throughput, one main challenge while pipelining and forwarding interests to multiple paths is to ensure that linearly independent segments are sent as replies. We impose constraints called MILIC (explained in Section 3.2) on the possible coded segments that can satisfy an interest with index $i \in \{1, \ldots, n\}$: their encoding vector must follow constraints depending on i. MICN (through these constraints) ensures that coded segments sent as replies to interests with different indexes are always linearly independent.

In order to complement the design choices, some additional features are adapted. Interest cancellation is introduced as an option to cancel unnecessary interests that may be lingering in the network after the retrieval of required content at the clients (see Section 4.8.2). Just-in-time re-encoding is adapted to prevent content from becoming non-innovative while being queued (Section 4.8.1). When there are multiple paths to a node, the fastest path is preferred. However, with pipelining, the contents tend to queue on reverse fastest path. Content redirection allows all possible paths to be used (instead of just the fastest/ shortest path).

3.2 MILIC

The main idea of MILIC starts from the pipelining idea, where several interests for content generation g are sent in parallel, with the goal that each of them brings innovative coded segments.

For a generation of size n, we propose to use n distinct interests. Interest $i \in \{1, \ldots, n\}$ can be satisfied by any coded segment whose encoding vector belongs to a predefined subset $\mathcal{A}_i$ of the set of all possible encoding vectors. In the following, these constraints will be such that the set of all non-zero encoding vectors is partitioned into n non-overlapping subsets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ satisfying some additional constraints.

To ensure that the answer to any of the n interests is linearly independent of the other answers, the subsets must satisfy the following property.

Property 1. *For any $a_1 \in \mathcal{A}_1, \ldots, a_n \in \mathcal{A}_n$, the vectors $a_1, \ldots, a_n$ should be linearly independent, for any set of coefficients $(\alpha_1 \ldots \alpha_n) \in \mathbb{F}_q^n$, $\sum_{i=1}^{n} \alpha_i a_i = 0$ iff $\alpha_1 = \cdots = \alpha_n = 0$.*

An additional condition can be imposed on subsets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ to benefit from the observation that when a node sends the same interests over ℓ faces, ℓ answers to each of these interests will likely be received. Ideally, these replies should be linearly independent. This leads to a property of subsets that is not mandatory but desirable to improve the efficiency of the proposed solution.

Property 2. *Consider k distinct subsets $\mathcal{A}_{\pi(1)}, \ldots, \mathcal{A}_{\pi(k)}$ where π is a permutation of the integers 1 to k. Consider $\ell \geqslant 1$ vectors $a_\kappa^1, \ldots, a_\kappa^\ell$ chosen uniformly at random from each subset $\mathcal{A}_{\pi(\kappa)}$, $\kappa = 1, \ldots, k$ such that $\ell k \leqslant n$, then rank $\left(a_1^1, \ldots, a_k^\ell\right) = \ell k$ with high probability.*

Finally, one may try to exploit the fact that segments are coded with possible re-encoding at intermediate nodes. Intermediate nodes may have received several segments with coding vectors belonging to the same subset. It may be of interest to combine these to generate a coded segment with encoding vector belonging to another subset to satisfy interests for that subset. This translates into the following additional desirable property for the subsets.

Property 3. *Consider the subset $\mathcal{A}_i$, $i \in \{1, \ldots, n-1\}$. For any pair $\left(a_i^1, a_i^2\right)$ of linearly independent vectors belonging to $\mathcal{A}_i$, then with high probability, there exist $\alpha_1 \in \mathbb{F}_q^*$ and $\alpha_2 \in \mathbb{F}_q^*$ such that $\alpha_1 a_i^1 + \alpha_2 a_i^2 \in \mathcal{A}_k$ with $k \neq i$.*

3.2.1 Proposed construction

Here we propose a construction of the sets $\mathcal{A}_1, \ldots, \mathcal{A}_n$, called MILIC, that partly satisfies the above properties. Consider

$$\mathcal{A}_i = \left\{(v_1, \ldots, v_n) \in \mathbb{F}_q^n \,|\, v_i \neq 0 \text{ and } \forall j < i, v_j = 0\right\}, \tag{1}$$

with $i = 1, \ldots, n$. With this construction the sets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ form a partition of $\mathbb{F}_q^n \setminus \{(0, \ldots, 0)\}$. The cardinality of $\mathcal{A}_k$ is

$$|\mathcal{A}_k| = (q-1)\, q^{n-k}. \tag{2}$$

Thus, with the construction (1), the cardinality is decreasing with k. This implicitly imposes an ordering among sets.

We now describe how Properties 1, 2, and 3 are satisfied by MILIC (proofs are in A). Property 1 is satisfied by construction: consider any $a_1 \in \mathcal{A}_1, \ldots, a_n \in \mathcal{A}_n$. The matrix whose rows are $a_1, \ldots, a_n$ is in row echelon form, and thus of full rank. The vectors $a_1, \ldots, a_n$ are thus linearly independent.

Property 3 is proven for MILIC construction in A.1, illustrating that with MILIC two linearly independent coded segments that are replies to the same interest I_i can be used to generate a reply for interest I_k with $k > 0$.

Lemma 1 illustrates Property 2 for a single subset $\mathcal{A}_k$ provided that $\ell \leqslant n - k + 1$, and evaluates the probability of having rank $\left(a_k^1, \ldots, a_k^\ell\right) = \ell$ and Lemma 2 illustrates Property 2 for the k first subsets $\mathcal{A}_1, \ldots, \mathcal{A}_k$.

Lemma 1. *Consider ℓ vectors $a_k^1, \ldots, a_k^\ell$ chosen uniformly at random from the set $\mathcal{A}_k$, $k = 1, \ldots, n$, and with $1 \leq \ell \leq n$. The probability that $a_k^1, \ldots, a_k^\ell$ are linearly independent is*

$$\Pr\left(\textit{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell\right) = \prod_{j=1}^{\ell} \left(1 - \frac{q^{\ell-1} - 1}{(q-1)\, q^{n-k}}\right).$$

Proof. See A.2. □

Example 1. *Illustrating Property 2 for one subset, Table 1 provides $P_F(\ell, 1) = 1 - \Pr\left(\textit{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell\right)$ for vectors of $n = 10$ elements in $\mathbb{F}_{256}$ for different subsets $\mathcal{A}_k$ and different values of ℓ. One observes that choosing 5 vectors at random from any of the subsets $\mathcal{A}_k$, $k = 1, \ldots, 5$, results in a very high probability of getting linearly independent vectors. Consequently, if a client sends 5 interest packets for elements in $\mathcal{A}_k$ over different faces, it is likely, provided that these interests follow different paths leading to different sources or caches with independent content in the network, to get 5 linearly independent data packets.*

	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$
$\mathcal{A}_1$	0	2.11×10^{-22}	5.46×10^{-20}	1.39×10^{-17}	3.58×10^{-15}
$\mathcal{A}_2$	0	5.46×10^{-20}	1.39×10^{-17}	3.58×10^{-15}	9.16×10^{-13}
$\mathcal{A}_3$	0	1.39×10^{-17}	3.58×10^{-15}	9.16×10^{-13}	2.34×10^{-10}
$\mathcal{A}_4$	0	3.58×10^{-15}	9.16×10^{-13}	2.34×10^{-10}	6.01×10^{-8}
$\mathcal{A}_5$	0	9.16×10^{-13}	2.34×10^{-10}	6.01×10^{-8}	1.53×10^{-5}

Table 1: Probability of getting linearly dependent coded vectors chosen at random from $\mathcal{A}_k \subset \mathbb{F}_{256}^{10}$

Lemma 2. *Consider $\ell \geqslant 1$ vectors $a_\kappa^1, \ldots, a_\kappa^\ell$ chosen uniformly at random from each subset $\mathcal{A}_\kappa$, $\kappa = 1, \ldots, k$ such that $\ell k \leqslant n$. The probability that $a_1^1, \ldots, a_k^\ell$ are linearly independent is*

$$\Pr\left(\textit{rank}\left(a_1^1, \ldots, a_k^\ell\right) = \ell k\right) = \prod_{j=1}^{(\ell-1)k} \left(1 - \frac{q^{j-1}}{q^{n-k}}\right).$$

Proof. See A.2.

Notice that the probability is close to one for appropriate parameter choices, see the following example. □

Example 2. *Table 2 provides* $P_F(\ell, k) = 1 - \Pr\left(\textit{rank}\left(a_1^1, \ldots, a_k^\ell\right) = \ell k\right)$ *for vectors of a generation of size* n *in* $\mathbb{F}_q$ *when choosing at random* ℓ *vectors from each subset* $\mathcal{A}_\kappa$, $\kappa = 1, \ldots, k$. *For* $n = 100$, *one observes that when a node receives* 2 *random packets from each* $\mathcal{A}_\kappa$, $\kappa = 1, \ldots, 50$ *subsets, provided that NC is performed in* $\mathbb{F}_{256}$, *the probability of getting a linearly independent packet is above* 99.6%. *The same result is obtained when* 4 *packets are obtained from each of the* $k = 25$ *first subsets. The constraints introduced by the subsets do not degrade the generation recovery performance significantly compared to plain NC. This result is mainly obtained due to the fact that one considers packets received from the first (largest) subsets.*

k	ℓ	$\mathbb{F}_2$	$\mathbb{F}_{256}$
50	2	0.71	0.0039
25	4	0.71	0.0039
33	3	0.42	1.53×10^{-5}
49	2	0.23	5.98×10^{-8}
48	2	0.06	9.13×10^{-13}
32	3	0.06	9.13×10^{-13}
24	4	0.06	9.13×10^{-13}
47	2	0.015	1.39×10^{-17}
45	2	0.00097	3.24×10^{-27}

Table 2: Probability $P_{\mathrm{F}}(\ell, k)$ of getting ℓ linearly dependent vectors chosen at random from consecutive subsets $\mathcal{A}_1$ to $\mathcal{A}_k$

Remark 1. *The size of the subsets* $\mathcal{A}_k$ *decreases when* k *increases, see (2). Thus, at first sight, considering the size of the subsets, given a set of random vectors (*e.g.*, from the cache of a router), there would be more possibilities to generate a coded segment with an encoding vector in the first subsets (larger) than in the last subsets (smaller). Nevertheless, due to pipelining behavior, this is not a problem. When there is a single path between a client and a source, Property 1 ensures that all contents are innovative. If* ℓ *distinct paths connect the client to one or* ℓ *sources or independent caches, the first interest packet in the pipeline should bring back* ℓ *linearly independent data packets thanks to Property 2. Then again, thanks to Property 2, the* k *first pipelined interest packets are likely to bring back* $k\ell$ *linearly independent data packets, see Table 2. Consequently, when* ℓ *distinct paths connect the client to one or several sources, it is unlikely that this client will need to send interests for contents in the subsets* $\mathcal{A}_k$ *with* k *close to* n. *This opens the potential for an adjustment and optimization of the size of the pipeline.*

4 MICN Protocol

This section complements the description of the MICN architecture introduced in Section 3.1. It focuses on the interest and content processing using the MILIC construction presented in Section 3.2, to recover linearly independent content with each interest in the context of NDN.

4.1 Content Segmentation and Naming

The original content C is partitioned into G smaller groups of segments, called *generations* $C = [c_1, c_2, ..., c_G]$. Each generation c_g, $g = 1, \ldots, G$ contains n *equally-sized* segments $c_g = [c_{g,1}, c_{g,2}, ..., c_{g,n}]$. The NC operations are restricted to segments that belong to the same generation and are assumed to be performed in $\mathbb{F}_q$.

A MILIC-compliant coded segment, with encoding vector in the subset $\mathcal{A}_i$, $i = 1, \ldots, n$, is defined as

$$\widetilde{c}_{g,i} = \sum_{j=i}^{n} a_j c_{g,j}$$

with $a_i \in \mathbb{F}_q^*$. The entries of $c_{g,j}$, $j = 1, \ldots, n$ and $\widetilde{c}_{g,i}$ are represented as elements of $\mathbb{F}_q$. Any coded segment $\widetilde{c}_{g,i}$ is identified by a prefix, a generation ID g, a MILIC index i, and the encoding vector $a = (0, \ldots, 0, a_i, \ldots, a_n) \in \mathbb{F}_q^n$ to indicate the weight of each original segment in $\widetilde{c}_{g,i}$. Consequently, we propose to identify $\widetilde{c}_{g,i}$ by the NDN name `<prefix>/micn/<`g`>/<`i`>/<`$a_i, \ldots, a_n$`>` (`micn` indicates that the content is network coded). Other naming conventions are possible with MICN.

4.2 Requesting MILIC-compliant contents

According to the naming convention of content segments, see Section 4.1, the name carried by the interest $I_{g,i}$ for a coded segment from C belonging to the generation g and with an encoding vector in $\mathcal{A}_i$ is `<prefix>/micn/<g>/<i>`.

Contrary to other proposals integrating NC to NDN/CCN, this interest format allows the client nodes to pipeline multiple interests for the same generation, provided that different $\mathcal{A}_i$ are specified in the names.

In practice, a client starts sending successive interests for coded segments in a given generation g, starting from packets with encoding vectors in $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_\rho$, where ρ is the pipeline size. Additional interests are sent once the content starts flowing back. The pipeline size ρ limits the number of outstanding interests from a client node at any time.

Each interest $I_{g,i}$ has an associated time-out. If no innovative content in response to $I_{g,i}$ is received before time-out, $I_{g,i}$ is sent again. Time-out may occur, *e.g.*, in case of losses of the interest or data packets.

4.3 MICN-compliant PIT

Compared to the classical NDN PIT, a MICN-compliant PIT identifies interests requesting coded segments with the same prefix and generation ID as *related interests.* PIT entries for related interests are grouped in a sub-table (identified by the prefix and generation ID g). Each entry itself includes the associated index i, *nonce* ν, as well as the *in* and *out* faces. The PIT entries are sorted by order of arrival.

Figure 3 illustrates a part of a MICN-compliant PIT at a given node with three faces f_1, f_2, and f_3. Three interests have been received and have been forwarded. The two interests associated with $\mathcal{A}_1$ are considered different since they have different nonces, which implies that different clients sent them. This is sufficient to implement the semantics that does not suppress the interests of different clients.

4.4 Just-in-Time Content Re-encoding / Replying

In plain NDN, whenever a node can satisfy an interest, a copy of the requested content is sent immediately. In MICN, as in some other NC-NDN protocols, nodes do not just forward a copy of the cached coded segment as an answer to the matching interests. They linearly combine cached segments from the same generation to generate a new coded segment.

PIT			
`<prefix>/micn/<g>`			
Index	Nonce	in-faces	out-faces
1	ν_1	f_1	f_2, f_3
1	ν_2	f_1	f_2, f_3
2	ν_3	f_2	f_1, f_3
$\vdots$	$\vdots$	$\vdots$	$\vdots$
k	ν_k	f_1	
`<prefix>/micn/<g'>`			

Figure 3: MICN compliant PIT: the interest with index k lead to cache hit and is temporarily stored in the PIT until the queue of face f_1 is empty to send back the associated data packet

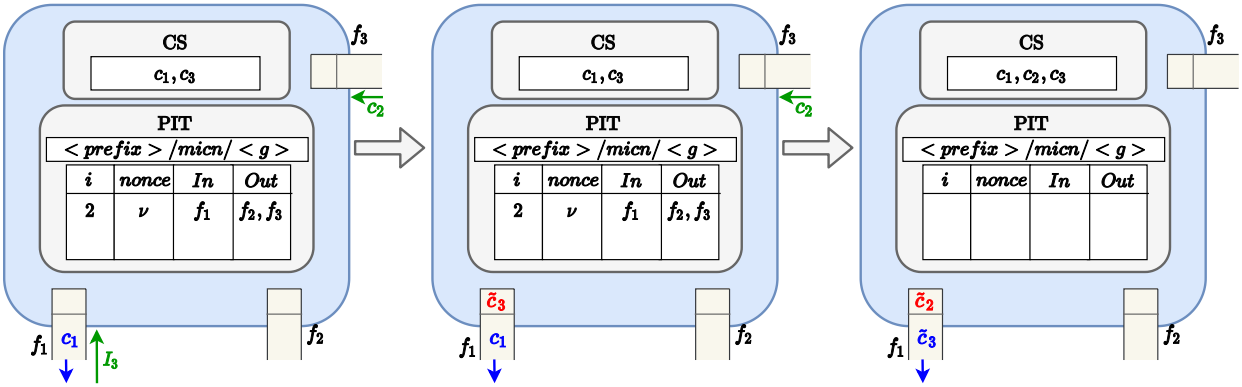
In MICN, the reply strategy is further modified, compared to plain NDN. A node waits until the queue of a face is empty before generating a coded segment that satisfies a pending interest on this face. This allows the node to use its latest cached contents when replying, hence sending more diverse content through the network. To achieve this, one packet queue is considered at the faces that is filled only when the packet in transit is completely delivered. The process to achieve this just-in-time re-encoding is detailed in Sections 4.5 and 4.7.

4.5 Interest processing

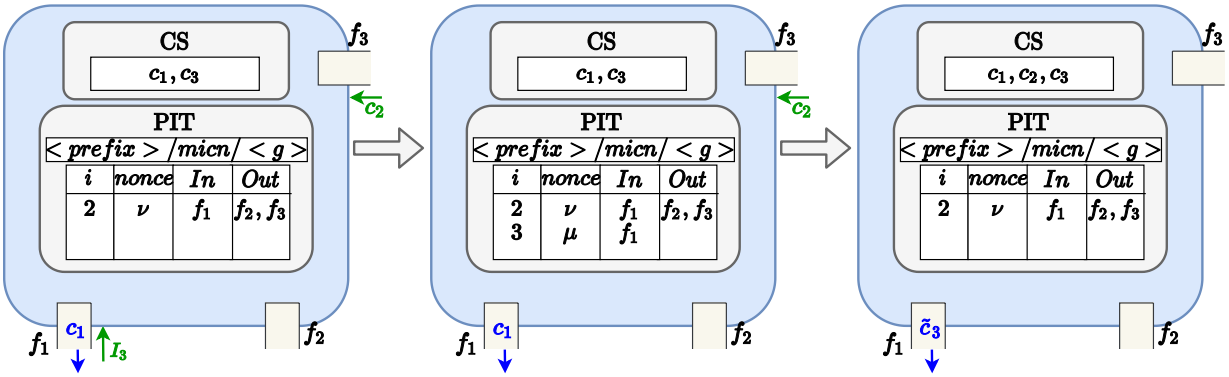
When a node receives an interest $I_{g,i}$, it initially performs loop detection. If an interest with the same nonce has already been received, $I_{g,i}$ is considered a looping interest. Otherwise, the node can either reply using a content generated from its CS or further forward the interest to the network.

4.5.1 CS lookup

Like the PIT, the *related contents* (*i.e.*, contents with the same prefix and generation ID) are grouped in the CS. The CS can store related coded segments with NC vectors in row echelon form. The *CS lookup* starts by identifying the related content matching the received interest. A *cache hit* occurs if there exists a coded segment belonging to the subset requested in the interest. This coded segment or all linear combinations of this segment with segments from subsets with a higher index can satisfy this interest. Later, when generating a reply, a variant of RLNC can be used.



(a) Immediate re-encoding: $\widetilde{c}_3 = \alpha_1 c_1 + \alpha_3 c_3$ is put in the outgoing queue of face f_1 before the reception and processing of content packet c_2.



(b) Just-in-time re-encoding with MICN: $\widetilde{c}_3 = \alpha_1 c_1 + \alpha_2 c_2 + \alpha_3 c_3$ is put in the outgoing queue of face f_1 only once this queue is empty; this gives the opportunity to the later received c_2 on face f_2 to be included in $\widetilde{c}_3$.

Figure 4: Re-encoding cached content

In case of a cache hit, the node schedules a reply for the interest. The node first checks the outgoing queue of the face where the interest arrived. If the queue is empty, the content is immediately sent in a data packet. Otherwise, a reply (linear combination) is generated only when the queue becomes empty. In our implementation, this scheduling is achieved by creating a *volatile* PIT entry to store the incoming face, nonce, *etc.*, but without specifying an outgoing face, since the interest does not require to be forwarded. See, for example, the interest with index k in Fig. 3.

Fig. 4a illustrates a node without just-in-time re-encoding. When it has enough content in its CS to respond to the incoming interest I_3. It immediately uses the related cached content to generate a response $\widetilde{c}_3$. However, the content remains in the queue until the content c_1 is transmitted. While the MICN node in Fig. 4b waits until c_1 is transmitted since it may receive more content and have a more diverse CS (since more content from the same generation is requested). So a volatile PIT entry is generated that is replied as soon as the queue becomes empty.

4.6 Interest Forwarding

In case of a cache miss, the node forwards the interest to its next hop neighbors on available faces in the FIB (except the incoming face) and creates a PIT entry, which records the incoming and outgoing faces. Unlike classical NDN, different nonces result in different entries, see Fig. 3.

Regarding the FIB management, multiple interest forwarding strategies can be implemented depending on the subset of chosen faces to forward the content. In this paper, to take advantage of the multiple paths to the source(s) and to have the opportunity to receive multiple linearly independent segments, the FIB is filled with all faces that can lead to a source *without looping back* to the node. The multicast forwarding strategy is then used, in which interests are forwarded on all faces in the FIB, as suggested in [4, Section 5.2.2].

4.7 Content Processing

When a coded segment arrives at a node, it is added in the CS if linearly independent with the already cached related segments. If the encoding vector belongs to a subset not already in the CS, it is immediately added in CS (linear independence guaranteed by Property 1). If the received coded segment has a NC vector belonging to the same set as a segment in the CS, partial Gaussian elimination is performed to verify linear independence before adding it to the CS. The updated cache might then satisfy some additional/new interests.

The node then uses its updated cache to reply to the pending interests. Whenever the queue of a face is empty, the node checks if any pending interest on that face can be satisfied utilizing the current state of the cache. It answers the oldest PIT entry, that may be satisfied and removes the entry.

4.8 Optimizations

In this section, we introduce some optimization compared to the classical NDN to improve the performance of MICN in an NC-NDN scenario.

4.8.1 Content Redirection

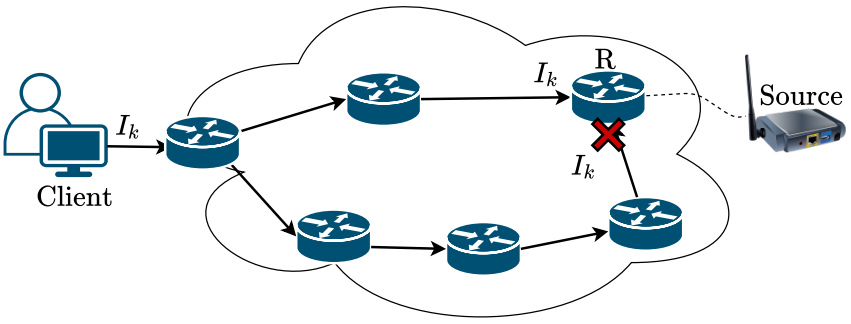


Figure 5: Content redirection scenario

A node can receive an interest on an alternate face while the same interest (same nonce) is still pending at the first face due to the Just-in-time content re-encoding of MICN, see Section 4.4. The new interest carrying the same nonce is considered looping and ignored. A common occurrence is shown in Fig. 5 where one path is faster (*e.g.*, shortest) than the other. All interests and, ultimately, contents will queue on the fastest past while the slow path will not be used.

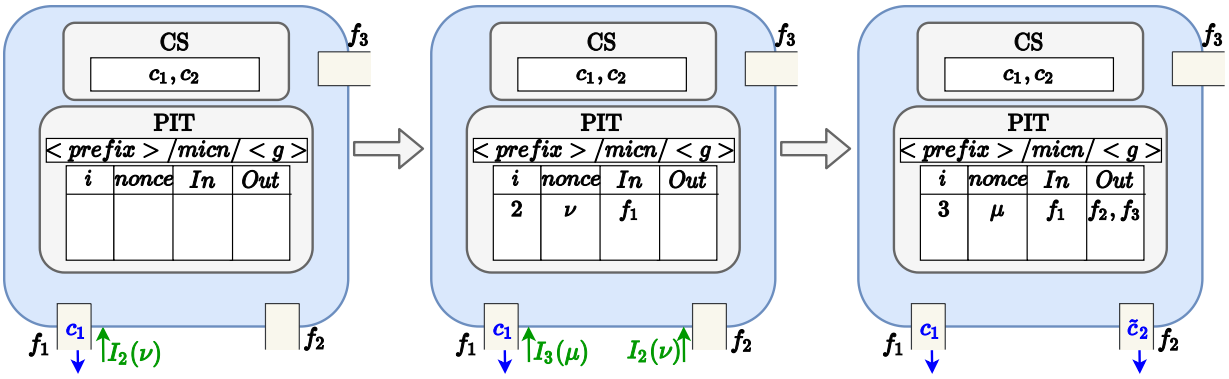


Figure 6: Content redirection on face f_2: during the transmission of c_1, an interest for content associated to $\mathcal{A}_2$ has been received from face f_1 (left) and then from face f_2 (middle); since the outgoing queue of face f_1 is still occupied, $\tilde{c}_2$ is transmitted on face f_2 (right).

Content redirection attempts to utilize all available paths by exploiting the information brought by the redundant interest that there exists an alternate path to the client. Suppose the queue associated with this alternate/second face is empty, and the node has matching content, it is immediately redirected to the client via this alternate face. This redirection is likely to improve the network utilization by benefiting from all paths leading to the client. In case of Fig. 5, both links should be fully utilized.

Fig. 6 depicts the state of a node that receives interest I_3 with the same nonce ν from an alternate face f_2 with an empty queue. Since the node has enough content to generate a reply for the interest but the face f_1 is busy, the node redirects the content via the alternate face to immediately send the reply and possibly benefit from a second path to the client.

4.8.2 Interest Cancellation (MICN-IC)

We observe that content continues to flow in the network due to delay and connectivity differences in different parts of the network even after the client nodes have received enough content to decode a generation. In order to reduce the traffic represented by redundant contents, we introduce the concept of *interest cancellation* (IC).

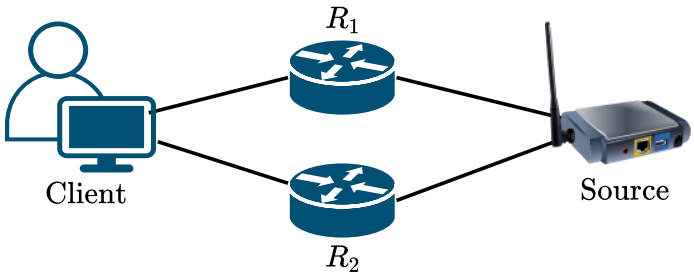


Figure 7: Topology where interest cancellation may be useful

Fig. 7 shows the simplest scenario justifying interest cancellation. There are 2 paths between the client and the source and, interests will be forwarded on both paths. The client will retrieve content at a rate twice faster than the routers R_1 and R_2. When the transfer of content completes at the client, there will still be pending interests at the intermediate routers. The content will still flow for these interests; this content is redundant for the clients. IC consists of having intermediate nodes deleting lingering interests by using information carried by the new interests on the content that is already retrieved at the client.

The optional *client identifier* and *state* fields are added in interest packets to perform cancellation. The client identifier field can be any unique node identifier[2]. The state field bears the information of subsets as defined by MILIC for which

[2]hash of the client node identifier or a randomly generated client nonce

that client has already available content. Such content may have been directly obtained or deduced after Gaussian elimination involving several data segments. The state field can be represented by a bitmap indicating the available indices. Notice that client identifier and state fields introduce a small overhead on interest packets.

When receiving interest with the state of a client, a node may ignore the pending interests referencing subsets $\mathcal{A}_i$ for the indices i for which content is already available. Nevertheless, the nodes do not immediately delete them: instead, they get low priority for replies, contrary to other related interests in the PIT, which have a normal priority. Due to Properties 2 and 3, answering the low-priority interests may still be useful: NC segments sent as replies even for subsets from which compliant content is already available at the client may bring new information with a high probability. Thus, a router first replies to interests that are guaranteed to increase the rank of the client and then to interests that are very likely to increase it.

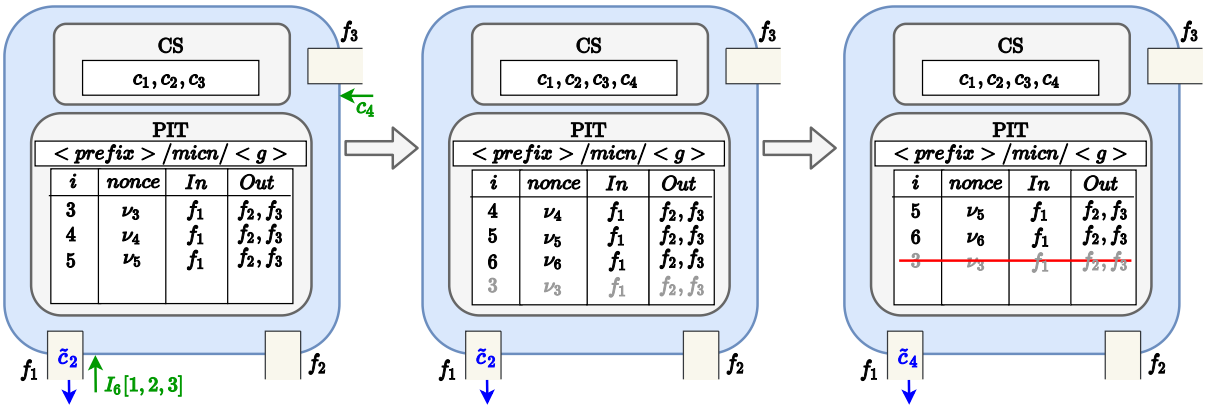


Figure 8: Interest Cancellation: An interest for packets associated to $\mathcal{A}_6$ is coming from face f_1, indicating that the source has already access to content associated to $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ (left); The pending interest for content associated to $\mathcal{A}_3$ is first tagged with low priority (middle); This pending interest is canceled as soon as an interest associated to a subset of higher index (here $\mathcal{A}_4$) is replied to (right).

A reply to interest with a low priority is sent only if the outgoing face is empty, and the node cannot generate content for interests with a normal priority. The deletion of low-priority interests occurs when the node has sent content for an interest with a higher index to the client. This version of MICN with Interest Cancellation is referred to as MICN-IC.

Fig. 8 illustrates the state of a node that receives interest for some content associated to $\mathcal{A}_6$. The interest also carries the state of the requesting node indicating, that it has already access to contents associated to $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$. The node sets the pending interest for content associated to $\mathcal{A}_3$ to low priority (interest in gray) using this information. This low priority interest is deleted once a content associated with $\mathcal{A}_4$ has been sent to the considered client. Without interest cancellation the node would continues to process interests and ensures that all pending interests are responded to in FIFO order based on the available content in the CS.

4.9 Summary of MICN Messages and Data-Structures

Table 8 compares MICN with classical NDN and NetCodCCN in terms of information stored in interest and data packets as well as that in routers. It gives an overview of the necessary information for the semantics of each protocol. For a given setting, it could also be the basis of a precise computation of each protocol's packet header size. In many cases, the NC vector, whose size increase linearly with generation size, will dominate the overhead, for instance, for generation size 100 and $\mathbb{F}_{2^8}$.

Interest Packet	
NDN	prefix, segment-id
NetcodCCN	prefix, gen-id
MICN	prefix, gen-id, MILIC-index
MICN-IC	prefix, gen-id, MILIC-index, client-id, bitmap (state)
Content Packet	
NDN	prefix, segment-id, content segment
NetcodCCN	prefix, gen-id, NC vector, coded segment
MICN	prefix, gen-id, MILIC-index, NC vector, coded segment
MICN-IC	prefix, gen-id, MILIC-index, NC vector, coded segment
PIT	
NDN	prefix, in-faces, out-faces, nonce-list
NetcodCCN	prefix, in-faces, out-faces, content counters[3]
MICN	prefix, in-face, out-faces, nonce[4]
MICN-IC	prefix, in-face, out-faces, nonce[4], priority

Table 3: Information in interest and data packets as well as that stored in routers (gen-id is generation identifier)

5 Evaluation

5.1 Simulation setup

We implemented our simulator in Python. The simulator includes a generic packet network simulator (scheduler, link, packet transmission), on top of which we developed an implementation of the proposed MICN protocol. We also did lightweight reimplementations of NDN and NetCodCCN, capturing the main semantics of these protocols. This includes all the semantics described in Section 4.6 and the data structures, PIT, FIB, and CS, along with the interest forwarding (with suppression, management of similar interests, multicast forwarding) in the spirit of [4]. For NetCodCCN, the main part is interest processing, forwarding and we implemented the semantics as described in [22] (including Algorithm 1 and 2)[5].

We compared the protocols over a simple butterfly topology and a more elaborate topology close to the PlanetLab topology from NetCodCCN [22]. At the link level, the parameters of our simulations are a propagation delay of 0.1 time unit for each packet, a transmission time of 1 time unit for data packets, and a very small transmission delay for interest packets ($1/(10 \times 2^{14}) \simeq 6 \times 10^{-6}$). A small amount of uniformly distributed transmission jitter (between 0 and $(1/10 \times 2^{21}) \simeq 3.8 \times 10^{-7}$) was also introduced.

In each topology, we consider the following scenario. Several clients request coded content, divided into generations of 100 segments each. We study the transmission of one generation. Each source stores a complete set of 100 segments. We assume that the intermediate nodes have enough cache space to store all segments of a generation. All the coding operations are performed in the finite field $\mathbb{F}_{2^8}$.

An interest pipeline size $\rho = 10$ is considered, the FIB and interest forwarding are as described in Section 4.6. At the client, each interest packet has a time-out of 10 time units (*i.e.,* equivalent the transmission delay of 10 data packets, that is a bit longer than the longer round-trip delay). If a client does not receive innovative content for interest after this time interval, and the content has not yet been decoded, it will resend the interest.

For the results, the goal was to focus on the throughput of content. The performance is evaluated in terms of download time, *i.e.*, the time it takes for a client to download and decode a generation. Since we ultimately focus on throughput, almost all delays come for content transmission (assuming a fixed size). All the header overheads are ignored (see Table 3, for the overhead comparison). Packet processing delays (coding/recoding delays), buffer limits are neglected. An upper bound of the throughput (content/time unit) received by a client is given by the maximum flow of the graph from the sources to the node. From this max-flow, one can derive a lower bound of the download time. In similar settings, it had been proven that NC could approach the max-flow bound [27], hence representing a meaningful benchmark. Another metric of interest is the total number of data packets exchanged in the network until all clients

[3] one counter per face for each prefix and generation id

[4] one entry per nonce

[5] Interest expiration was not implemented, since NetCodCCN assumes an expiration time large enough that any forwarded Interest will bring the requested segment before its expiration.

have retrieved the generation with no interest or data packet present in the network anymore. MICN and NetcodCCN use a multicast strategy; for fairness of comparison, a multicast strategy is also considered in NDN.

5.2 Results with the butterfly topology

We first analyze the behavior of MICN on a simple butterfly topology with two sources S_1 and S_2, and two clients U_1 and U_2, connected through a set of intermediate caching routers as represented in Fig. 9.

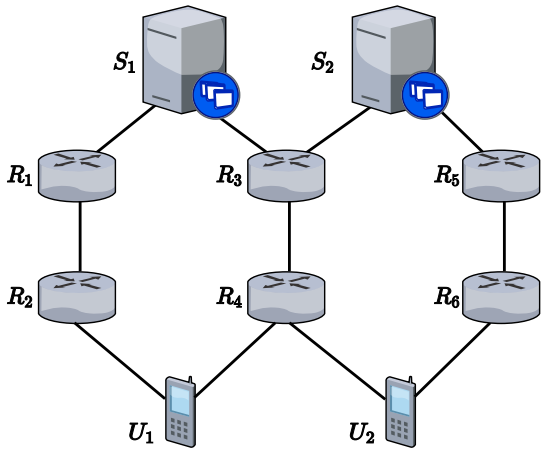
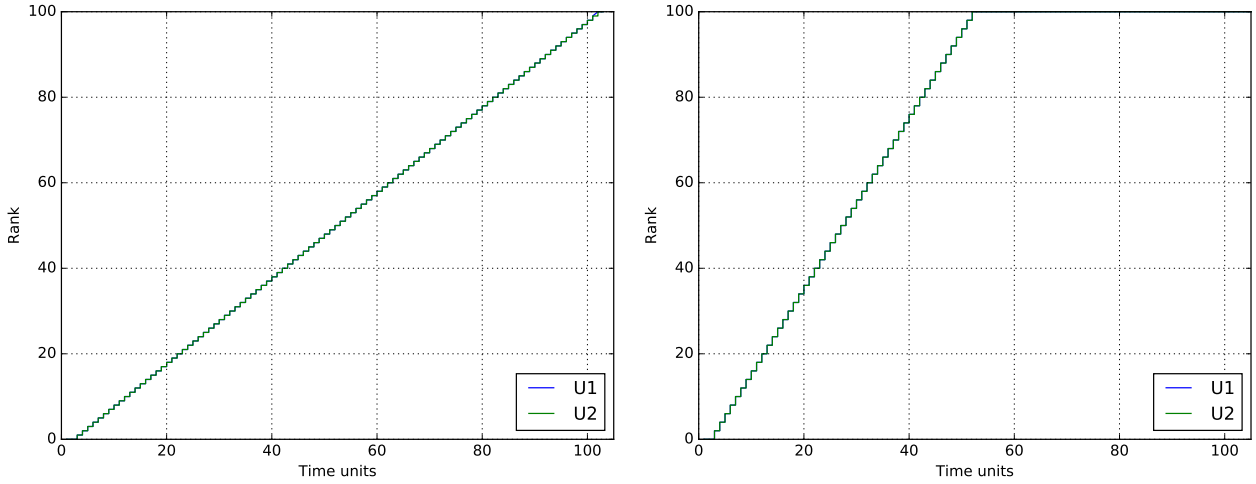


Figure 9: Butterfly topology

The performance of MICN mainly depends on how the bottleneck link ($R_3 \leftrightarrow R_4$) is used. With classical NDN, the two clients U_1 and U_2 should request precisely the same segments on the middle link to improve performance. Nevertheless, the clients would require topology knowledge and coordination to do so. However, this is not required with NC, and the clients can simultaneously send their interests to all their available faces.

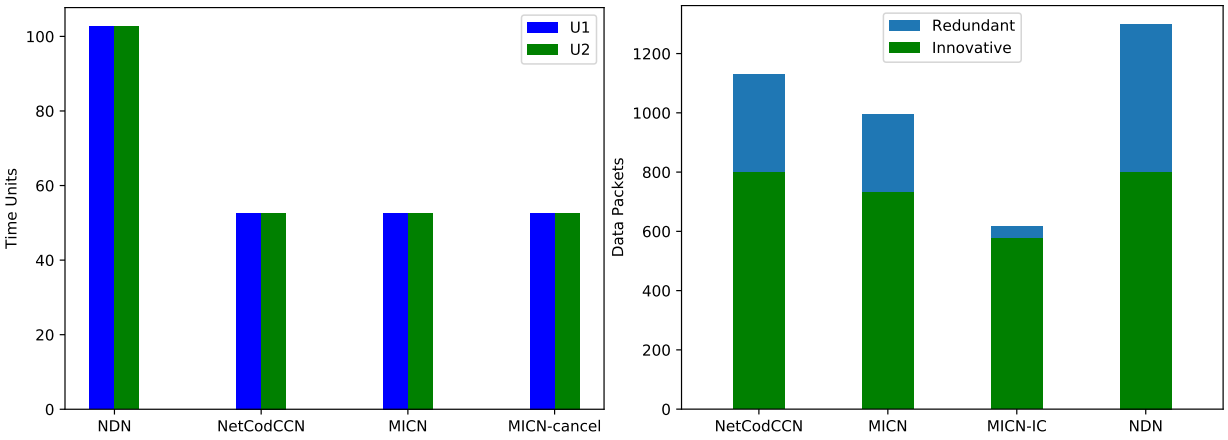
Fig. 10 shows the rank evolution of the client nodes over time for MICN, MICN-IC, NetCodCCN, and NDN. MICN, MICN-IC, and NetCodCCN retrieve content at the max-flow rate at each client, *i.e.*, each data packet received at the client is innovative. After some initial delay, due to propagation, the clients receive 2 linearly independent data packets every time unit, as shown in Fig. 10b. The results are for MICN, but the results are identical for NetCodCCN and MICN-IC, as confirmed by the final content retrieval time in Fig. 11a.



(a) NDN

(b) MICN, MICN-IC and NetCodCCN(all perform similarly)

Figure 10: Butterfly topology: rank evolution as a function of time

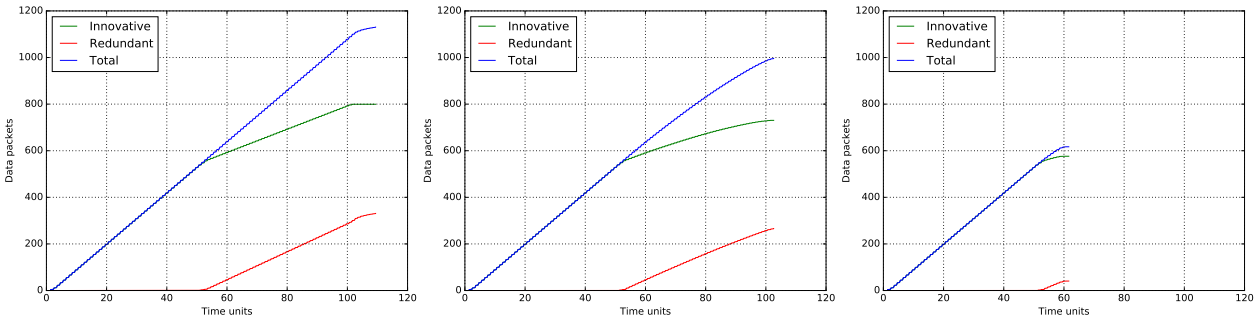


(a) Content retrieval time (b) Cumulative data traffic

Figure 11: Butterfly topology: comparison of protocol performance

Nevertheless, there are significant differences in the volume of data traffic that each protocol generates, as shown in Fig. 11b. Note that after NDN, NetcodCCN generates the most data traffic. MICN has a slightly reduced amount of data traffic, and MICN-IC has the least amount of traffic that mostly accounts for the innovative traffic.

To understand the data traffic flowing in the network, Fig. 12 depicts the evolution with time of the cumulative number of data packets transmitted on all the links of the network, counted from time $t = 0$. The curves end when transmission of data packets stops. In the beginning, there is only innovative traffic in the network, *i.e.*, all data packets are innovative for the intermediate nodes as well as the client node receiving them. Note that towards the end, when the clients have received the entire generation (around 52 time units), there is still data traffic flowing in the network. Since all the network nodes do not have the same min-cut, they receive content at different rates. The intermediate nodes with a lower min-cut than the clients (see Fig. 10) continue to get responses for interests that they have forwarded in the past. The content that arrives is still innovative for them. The intermediate nodes also continue to forward content to satisfy interests in their PIT (no longer innovative for the clients; hence the redundant traffic curve starts to grow).

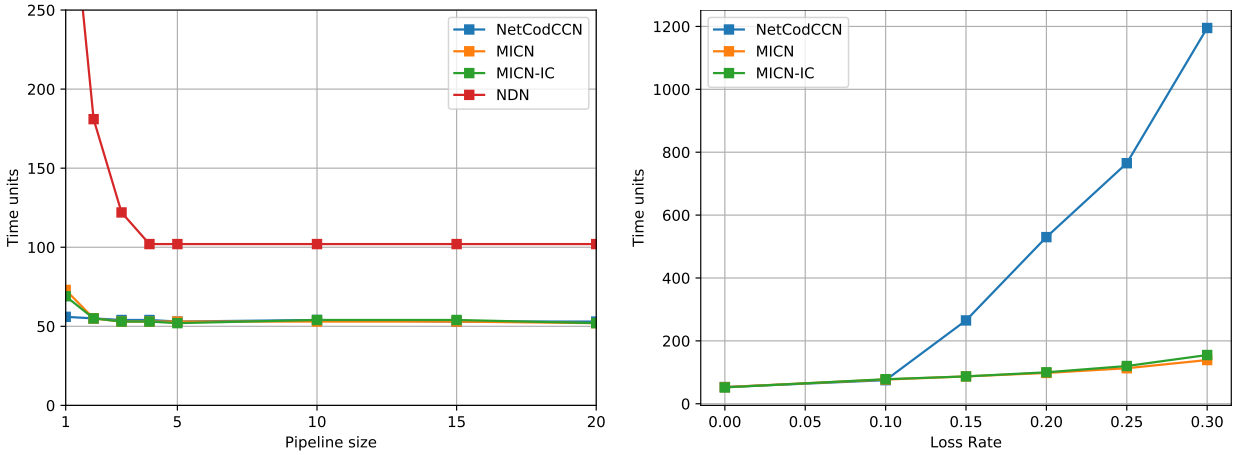


(a) NetCodCCN (b) MICN (c) MICN-IC

Figure 12: Butterfly topology: evolution of cumulative data traffic as a function of time

MICN-IC deletes interests tagged with low priority, pending even if a client has access to content for those interests. Canceling such interests reduces the redundant data traffic, at the price of some signaling overhead. Precisely, in butterfly topology (Fig. 9), 10 transmissions of data packets over various links are necessary for delivering 2 data packets to the clients U_1 and U_2, *i.e.*, 5 transmissions per packet. For a generation of size 100, a minimum of 500 transmissions are required for both clients to receive the entire generation. Fig. 12 shows that with MICN-IC, a slightly larger amount of transmissions are required. NetCodCCN achieves similar download performance, but interests are not canceled, and several data packets are redundant, leading to increased traffic.

The effect of sending consecutive interests by clients is analyzed in Fig. 13a. To have a continuous flow of content in the butterfly topology (in the absence of losses), the clients need to have at least two outstanding interests at any time (because there are two paths) and usually even more because of the propagation delays. In the case of plain NDN with multicast strategy, the link $R_3 \leftrightarrow R_4$ becomes a bottleneck due to the absence of coordination among the clients. Even when the pipeline size increases, the performance cannot reach the one obtained with NC. MICN, MICN-IC, and NetCodCCN, however, with a sufficient pipeline size (here as small as $\rho = 5$), can reach the maximum capacity.



(a) Impact of pipeline size (without losses)

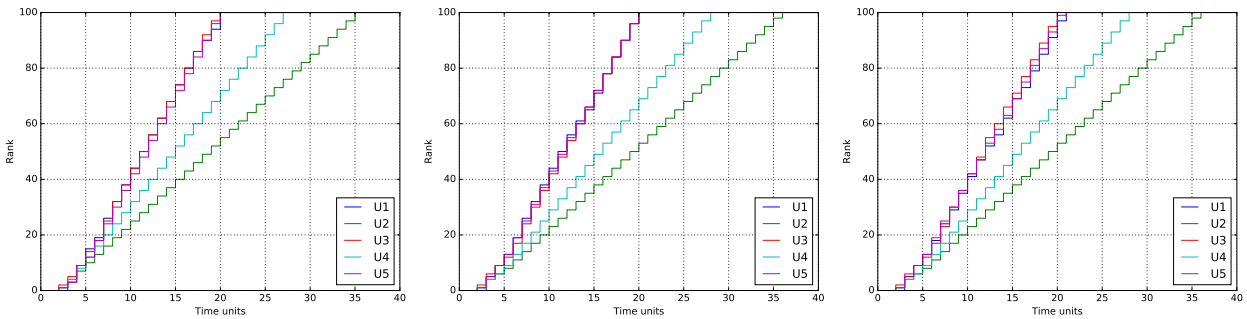
(b) Impact of transmission error rate

Figure 13: Butterfly topology: download time

Next, we evaluate the performance of MICN in case of losses. Fig. 13b, depicts the effect of losses on the performance of the protocols. We consider transmission losses modeled with a fixed loss probability for both interest and data packets[6]. MICN and MICN-IC appear to have much better performance compared to NetCodCCN. MICN has the advantage of precisely identifying which interest (pointing to a subset $\mathcal{A}_i$) has timed-out (no matching content received). In NetCodCCN, when a downstream data packet is lost, the router will consider the interest as satisfied (update its content counters). An interest repeated due to time-out is considered a new interest, and the router will typically forward it. In MICN, the repeated interest will be immediately satisfied by the router's cache.

5.3 Results with the PlanetLab topology

The behavior of MICN is then analyzed considering the PlanetLab topology from [22], with one source and five client nodes connected through a set of 20 intermediate caching routers.



(a) NetcodCCN

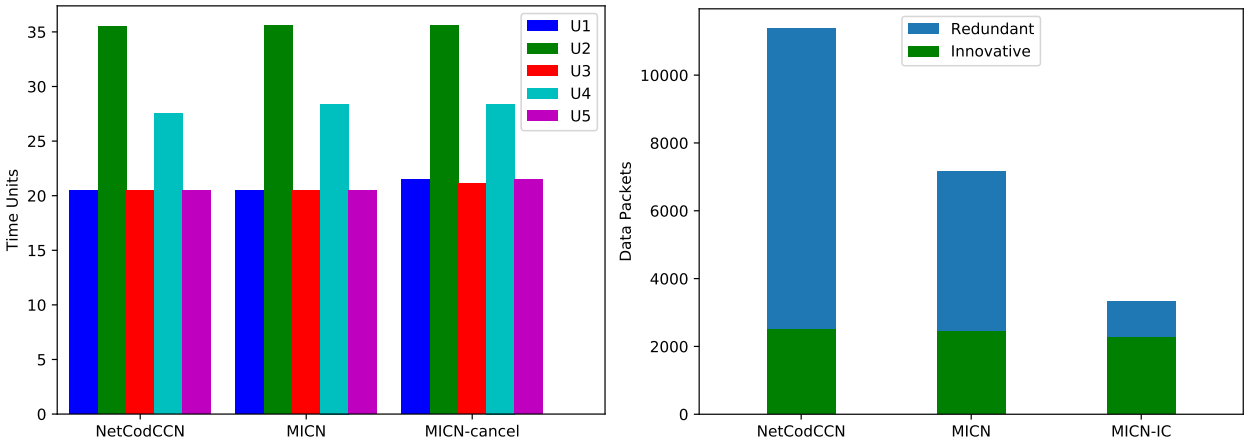
(b) MICN

(c) MICN-IC

Figure 14: PlanetLab topology: rank evolution as a function of time

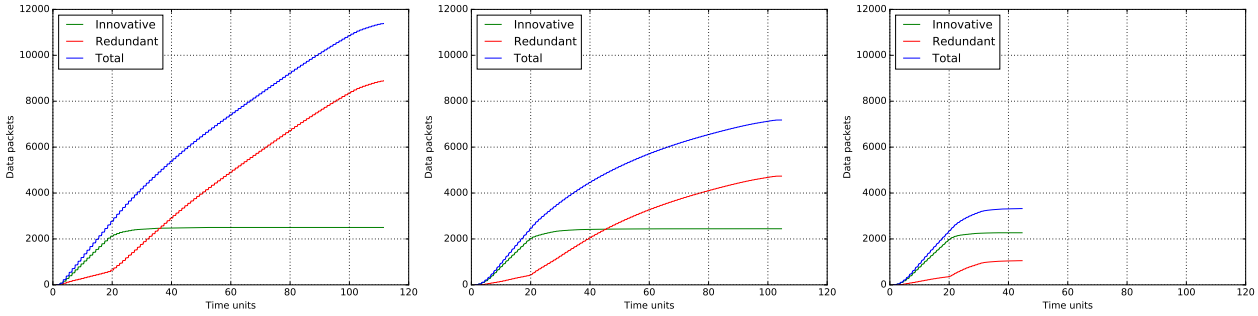
[6]Notice that NetCodCCN simulations in [22] consider only segment (data packets) losses. However, here both interest and data packets are prone to losses.

Fig. 14 shows the rank evolution of the client nodes over time for MICN, MICN-IC, and NetCodCCN. As seen in Fig. 15a, with MICN, MICN-IC, and NetCodCCN, clients receive enough content to decode a generation at a rate above 95% of the maximum rate (provided by the min-cut between the source and the clients), as observed for the butterfly topology. The difference between the data traffic generated by each network is represented in Fig. 15b. MICN has a better performance in terms of total traffic compared to NetCodCCN. MICN-cancel performs the best in terms of total traffic at the price of a slightly increased content retrieval time, due to cancellation.



(a) Content retrieval time (b) Cumulative data traffic

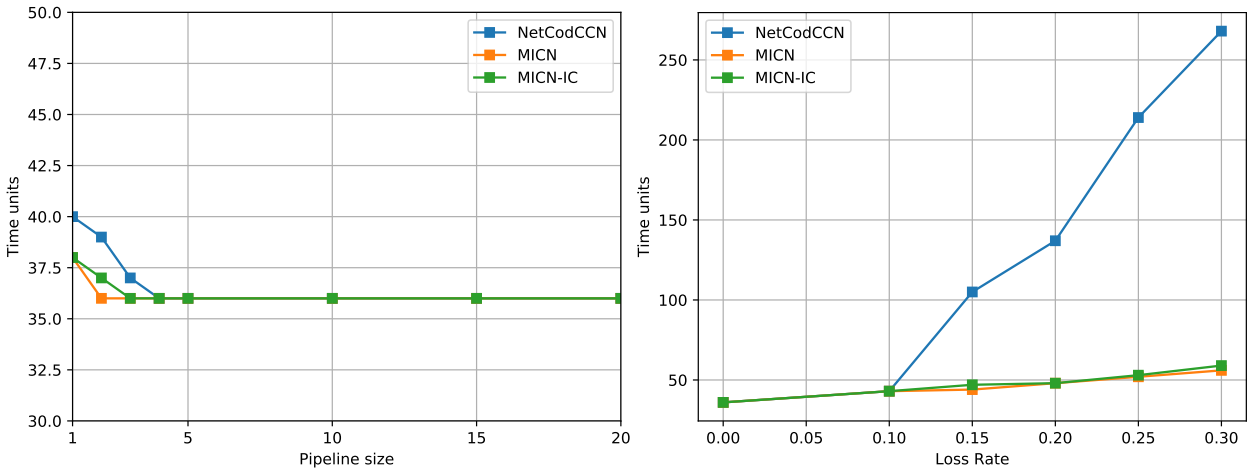
Figure 15: PlanetLab topology: comparison of protocol performance



(a) NetCodCCN (b) MICN (c) MICN-IC

Figure 16: PlanetLab topology: evolution of cumulative data traffic as a function of time

Fig. 16 illustrates the cumulative number of data packets transmitted on all network as a function of time. NetCodCCN generates the most data traffic (also for the longer duration). MICN is able to reduce the cumulative traffic by a considerable amount since the content does not have to be flooded on all the links as for NetCodCCN.MICN-IC performs the best in terms of traffic, with respectively 2.16 and 3.42 times fewer transmitted data packets compared to MICN and NetCodCCN. In the PlanetLab topology, with the considered scenario, the amount of non-innovative packets dominates: about 80% of the content traffic with NetCodCCN is redundant (non-innovative). Note that all the innovative traffic that appears in the cumulative traffic might not be useful for the clients because intermediate nodes of the network are unable to detect when a client has received all packets required to decode a generation.



(a) Impact of pipeline size (without losses)

(b) Impact of transmission error rate

Figure 17: PlanetLab topology: download time

In the PlanetLab topology, the pipeline size impacts the performance only when it is too small, as shown in Fig. 17a. Increasing the pipeline size above 2 (MICN), 3 (MICN-IC), and 5 (NetCodCCN) does not bring additional benefit.Fig. 17b shows the effect of transmission losses. The download time with MICN and MICN-IC increases almost linearly with the loss rate, compared to NetCodCCN, which increases faster when the loss rate is above 10%. In the PlanetLab topology, compared to the butterfly topology, MICN, MICN-IC, and NetCodCCN are all more robust to packet losses due to the more significant amount of redundant content traffic in the network, which helps to compensate for the losses.

6 Conclusion

In this work, we propose a novel way of integrating NC and interest-based ICN. The proposed MICN protocol is built around the MILIC construction that allows the clients to request content with encoding vectors that belong to predefined subsets by adding an index in the interest, that indicates the subset. This interest naming allows the nodes to send multiple interests in parallel and ensures that linearly independent content sent as reply. In the considered scenarios, the clients download content close to their maximum capacity (like NetCodCCN). Nevertheless, thanks to interest cancellation, MILIC-IC limits the redundant data traffic considerably. This reduces the network load and leaves free network resources to fetch content from consecutive generations.

Our future research includes investigating improved interest forwarding algorithms to use the multiple active links better and reduce the data traffic by adjusting the number of outgoing interests.

Acknowledgements

This research was partly supported by Labex DigiCosme (project ANR11 LABEX0045DIGICOSME) operated by ANR as part of the program *Investissement d'Avenir* Idex Paris-Saclay (ANR11IDEX000302).

References

[1] Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022, Cisco White Paper (2019).

[2] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos, A survey of information-centric networking research, IEEE Commun. Surveys Tuts. 16 (2) (2013) 1024–1049.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard, Networking named content, in: Proc. ACM CoNEXT, 2009, pp. 1–12.

[4] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. Dibenedetto, C. Fan, D. Pesavento, G. Grassi, G. Pau, H. Zhang, T. Song, H. B. Abraham, P. Crowley, S. O. Amin, V. Lehman, L. Wang, NFD Developer's Guide, Technical Report NDN-0021 Rev. 10 (July, 2018) (2018).

[5] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thorton, D. K. Smetters, B. Zhang, G. Tsudik, K. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, E. Yeh, Named Data Networking (NDN) Project, Technical Report NDN-0001 (October, 2010) (2010).

[6] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, TCP Extensions for Multipath Operation with Multiple Addresses, RFC 6824 (Jan. 2013).

[7] M.-J. Montpetit, C. Westphal, D. Trossen, Network coding meets information-centric networking: An architectural case for information dispersion through native network coding, in: Proc. ACM Workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications, 2012, pp. 31–36.

[8] R. Ahlswede, N. Cai, S.-Y. Li, R. W. Yeung, Network information flow, IEEE Trans. Inf. Theory 46 (4) (2000) 1204–1216.

[9] R. Koetter, M. Médard, An algebraic approach to network coding, IEEE/ACM Trans. Netw. 11 (5) (2003) 782–795.

[10] T. Ho, M. Medard, J. Shi, M. Effros, D. R. Karger, On randomized network coding, in: Proc. Annu. Allerton Conf. Commun Control Comput., Vol. 41, 2003, pp. 11–20.

[11] R. Ahmed, M. F. Bari, S. R. Chowdhury, M. G. Rabbani, R. Boutaba, B. Mathieu, αRoute: Routing on Names, IEEE/ACM Trans. Netw. 24 (5) (2016) 3070–3083.

[12] D. Posch, H. Hellwagner, B. Rainer, SAF: Stochastic adaptive forwarding in named data networking, IEEE/ACM Trans. Netw. 25 (2) (2017) 1089–1102.

[13] P. Gusev, J. Burke, NDN-RTC: Real-time videoconferencing over named data networking, in: Proc. ACM-ICN, 2015, pp. 117–126.

[14] D. Trossen, M. Sarela, K. Sollins, Arguments for an information-centric internetworking architecture, ACM SIGCOMM Comput. Commun. Rev. 40 (2) (2010) 26–33.

[15] K. Matsuzono, H. Asaeda, C. Westphal, Network Coding for Content-Centric Networking / Named Data Networking: Requirements and Challenges, Internet-Draft draft-irtf-nwcrg-nwc-ccn-reqs-04, Internet Engineering Task Force, work in Progress (Mar. 2020).

[16] P. A. Chou, Y. Wu, K. Jain, Practical network coding, Proc.Annu. Allerton Conf. Commun Control Comput. 41 (1) (2003) 40–49.

[17] K. Lei, S. Zhong, F. Zhu, K. Xu, H. Zhang, A NDN IoT Content Distribution Model with Network Coding Enhanced Forwarding Strategy for 5G, IEEE Trans. Ind. Informat. 14 (6) (2017) 2725–2735.

[18] Q. Wu, Z. Li, G. Xie, CodingCache: multipath-aware CCN cache with network coding, in: Proc. ACM SIGCOMM Workshop on ICN, 2013, pp. 41–42.

[19] Q. Wu, Z. Li, G. Tyson, S. Uhlig, M. A. Kaafar, G. Xie, Privacy-aware multipath video caching for content-centric networks, IEEE J. Sel. Areas Commun. 34 (8) (2016) 2219–2230.

[20] G. Zhang, Z. Xu, Combing CCN with network coding: An architectural perspective, Computer Networks 94 (2016) 219–230.

[21] Y. Liu, S. Z. Yu, Network coding-based multisource content delivery in Content Centric Networking, J. Netw. Comput. Appl. 64 (2016) 167–175.

[22] J. Saltarin, E. Bourtsoulatze, N. Thomos, T. Braun, NetCodCCN: A network coding approach for content-centric networks, in: Proc. IEEE INFOCOM, 2016, pp. 1–9.

[23] W. X. Liu, S. Z. Yu, G. Tan, J. Cai, Information-centric networking with built-in network coding to achieve multisource transmission at network-layer, Computer Networks 115 (2017) 110–128.

[24] K. Matsuzono, H. Asaeda, T. Turletti, Low latency low loss streaming using in-network coding and caching, in: Proc. IEEE INFOCOM, 2017, pp. 1–9.

[25] M. Bilal, S.-G. Kang, Network-Coding Approach for Information-Centric Networking, IEEE Syst. J. 13 (2) (2018) 1376–1385.

[26] J. Saltarin, E. Bourtsoulatze, N. Thomos, T. Braun, Adaptive Video Streaming With Network Coding Enabled Named Data Networking, IEEE Trans Multimedia 19 (10) (2017) 2182–2196.

[27] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, B. Leong, A random linear network coding approach to multicast, IEEE Trans. Inf. Theory 52 (10) (2006) 4413–4430.

A Proofs of MILIC properties

A.1 Proof of Property 3 for MILIC

To prove Property 3 for $k > i$, consider an intermediate node that received two linearly independent segments $a_i^1 \in \mathcal{A}_i$ and $a_i^2 \in \mathcal{A}_i$. The $i-1$ first entries of a_i^1 and a_i^2 are zero, and their i-th entries $a_{i,i}^1$ and $a_{i,i}^2$ are non-zero. Then, as $\mathbb{F}_q$ is a group for multiplication, considering any $\alpha_1 \in \mathbb{F}_q^*$, there exists $\alpha_2 \in \mathbb{F}_q^*$ such that $\alpha_1 a_{i,i}^1 + \alpha_2 a_{i,i}^2 = 0$. Moreover, since a_i^1 and a_i^2 are linearly independent, one has $b = \alpha_1 a_i^1 + \alpha_2 a_i^2 \neq 0$. Let k be the smallest index such that $b_k \neq 0$. Necessarily $k > i$ and $b \in \mathcal{A}_k$.

A.2 Proof of Property 2 for MILIC

We start proving Property 2 for a single subset $\mathcal{A}_k$ provided that $\ell \leqslant n-k+1$, evaluating the probability of having rank $\left(a_k^1, \ldots, a_k^\ell\right) = \ell$.

Lemma. 1 *Consider ℓ vectors $a_k^1, \ldots, a_k^\ell$ chosen uniformly at random from the set $\mathcal{A}_k$, $k = 1, \ldots, n$, and with $1 \leq \ell \leq n$. The probability that $a_k^1, \ldots, a_k^\ell$ are linearly independent is*

$$\Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell\right) = \prod_{j=1}^{\ell}\left(1 - \frac{q^{\ell-1}-1}{(q-1)\,q^{n-k}}\right).$$

Proof. Consider first $\ell = 2$, and $a_k^1 \in \mathcal{A}_k$. The set of non-zero vectors collinear to a_k^1 and included in $\mathcal{A}_k$ is span $\left(a_k^1\right) \cap \mathcal{A}_k = \text{span}\left(a_k^1\right) \setminus \{(0, \ldots, 0)\}$, whose size is $q-1$. When choosing a second vector $a_k^2 \in \mathcal{A}_k$ uniformly at random, the probability that a_k^1 and a_k^2 are linearly dependent is equal to the probability that $a_k^2 \in \text{span}\left(a_k^1\right) \setminus \{(0, \ldots, 0)\}$. Consequently, the probability that a_k^1 and a_k^2 are linearly independent is

$$\Pr\left(\text{rank}\left(a_k^1, a_k^2\right) = 2\right) = 1 - \frac{\left|\text{span}\left(a_k^1\right) \setminus \{(0, \ldots, 0)\}\right|}{|\mathcal{A}_k|} = 1 - \frac{1}{q^{n-k}}.$$

Assume now that the $j-1$ first vectors $a_k^1 \in \mathcal{A}_k, \ldots, a_k^{j-1} \in \mathcal{A}_k$ are linearly independent. The set of vectors that are linearly dependent with $a_k^1, \ldots, a_k^{j-1}$ and included in $\mathcal{A}_k$ is Span $\left(a_k^1, \ldots, a_k^{j-1}\right) \cap \mathcal{A}_k = \text{span}\left(a_k^1, \ldots, a_k^{j-1}\right) \setminus \{(0, \ldots, 0)\}$. Its size is $q^{j-1} - 1$. Then, when choosing $a_k^j \in \mathcal{A}_k$ uniformly at random, the probability that $a_k^1, \ldots, a_k^j$ are linearly dependent is equal to the probability that $a_k^j \in \text{span}\left(a_k^1, \ldots, a_k^{j-1}\right) \setminus \{(0, \ldots, 0)\}$. Consequently

$$\Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^j\right) = j \,|\, \text{rank}\left(a_k^1, \ldots, a_k^{j-1}\right) = j-1\right) = 1 - \frac{q^{j-1}-1}{(q-1)\,q^{n-k}}. \tag{3}$$

Then one has

$$\begin{aligned}
&\Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell\right) \\
&\qquad = \Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell, \text{rank}\left(a_k^1, \ldots, a_k^{\ell-1}\right) = \ell-1\right) \\
&\qquad = \Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell \,|\, \text{rank}\left(a_k^1, \ldots, a_k^{\ell-1}\right) = \ell-1\right) \\
&\qquad\qquad \Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^{\ell-1}\right) = \ell-1\right).
\end{aligned} \tag{4}$$

Applying this recursively and using (3), one gets

$$\begin{aligned}
&\Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell\right) \\
&\qquad = \prod_{j=2}^{\ell} \Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^j\right) = j \,|\, \text{rank}\left(a_k^1, \ldots, a_k^{j-1}\right) = j-1\right) \\
&\qquad\qquad \Pr\left(\text{rank}\left(a_k^1\right) = 1\right) \\
&\qquad = \prod_{j=1}^{\ell}\left(1 - \frac{q^{j-1}-1}{(q-1)\,q^{n-k}}\right).
\end{aligned}$$

□

We now prove Property 2 for the k first subsets $\mathcal{A}_1, \ldots, \mathcal{A}_k$.

Lemma. 2 *Consider $\ell \geqslant 1$ vectors $a_\kappa^1, \ldots, a_\kappa^\ell$ chosen uniformly at random from each subset $\mathcal{A}_\kappa$, $\kappa = 1, \ldots, k$ such that $\ell k \leqslant n$. The probability that $a_1^1, \ldots, a_k^\ell$ are linearly independent is*

$$\Pr\left(\textit{rank}\left(a_1^1, \ldots, a_k^\ell\right) = \ell k\right) = \prod_{j=1}^{(\ell-1)k} \left(1 - \frac{q^{j-1}}{q^{n-k}}\right).$$

Proof. According to Property 1, the vectors $a_1^1, \ldots, a_k^1$ are linearly independent. Consider the matrix A, whose first k rows are the vectors $a_1^1, \ldots, a_k^1$ and the $(\ell-1)\,k$ remaining rows are $a_\kappa^2, \ldots, a_\kappa^\ell$, $\kappa = 1, \ldots, k$. The first k rows are used to perform Gaussian elimination on the $(\ell-1)\,k$ remaining rows to get a matrix A_1 of the form

$$A_1 = \begin{bmatrix} 1 & * & \cdots & & & * \\ 0 & 1 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & & & \\ & & & 1 & * \cdots * & \\ & & & 0 & & \\ \vdots & & & \vdots & B & \\ 0 & \cdots & & 0 & & \end{bmatrix}.$$

In A_1, B is a matrix of $(\ell-1)\,k$ rows and $n-k$ columns. Since all vectors chosen in the subset $\mathcal{A}_\kappa$, $\kappa = 1, \ldots, k$, have been selected uniformly at random, the $n-k$ last entries of each vector are independently and uniformly distributed. The i-th row of B results in a linear combination of $a_1^1, \ldots, a_k^1$ with one of the remaining vectors $a_\kappa^2, \ldots, a_\kappa^\ell$, $\kappa = 1, \ldots, k$. Consequently, the $n-k$ components of the i-th row of B are still independently and uniformly distributed. Since all $n-k$ last components of $a_\kappa^2, \ldots, a_\kappa^\ell$, $\kappa = 1, \ldots, k$ are independently and uniformly distributed; all components of the matrix B are independently and uniformly distributed.

The matrix A is of full row rank ℓk iff the matrix B is full row rank $(\ell-1)\,k$. The first row $b_1 \in B$ is non-zero with probability $1 - \frac{1}{q^{n-k}}$. The second row $b_2 \in B$ has components that are uniformly and independently distributed from the other entries of B and thus of b_1. The vectors (b_1, b_2) are linearly independent if b_2 does not belong to the space spanned by b_1. Since span (b_1) is of size q, one has

$$\Pr\left(\text{rank}\left(b_1, b_2\right) = 2\right) = 1 - \frac{q}{q^{n-k}}.$$

Assume now that the $j-1$ first row vectors $b_1, \ldots, b_{j-1}$ of B are linearly independent. Under this assumption, the probability that b_j is such that the j first row vectors $b_1, \ldots, b_j$ of B are linearly independent is equal to the probability that b_j does not belong to the subspace of dimension q^{j-1} spanned by $b_1, \ldots, b_{j-1}$. Consequently,

$$\Pr\left(\text{rank}\left(b_1, \ldots, b_j\right) = j \,|\, \text{rank}\left(b_1, \ldots, b_{j-1}\right) = j-1\right) = 1 - \frac{q^{j-1}}{q^{n-k}}.$$

Then similarly as 4, the probability that B is of full rank is given by

$$\begin{aligned}\Pr\left(\text{rank}\left(B\right) = (\ell-1)\,k\right) &= \prod_{j=1}^{(\ell-1)k} \left(1 - \frac{q^{j-1}}{q^{n-k}}\right) = \prod_{j=1}^{(\ell-1)k} \left(1 - \frac{1}{q^{n-k-j+1}}\right). \\ &\approx 1 - \frac{1}{q^{n-lk+1}} \text{when } q \text{ large}\end{aligned}$$

□