Enhancing Multi-source Content Delivery in Content-Centric Networks with Fountain Coding

George Parisis
Univ. of Sussex, UK
g.parisis@sussex.ac.uk

Vasilis Sourlas,
UCL, UK
v.sourlas@ucl.ac.uk

Konstantinos V. Katsaros
UCL, UK
k.katsaros@ucl.ac.uk

Wei Koong Chai
UCL, UK
w.chai@ucl.ac.uk

George Pavlou
UCL, UK
g.pavlou@ucl.ac.uk

ABSTRACT

Fountain coding has been considered as especially suitable for lossy environments, such as wireless networks, as it provides redundancy while reducing coordination overheads between sender(s) and receiver(s). As such it presents beneficial properties for multi-source and/or multicast communication. In this paper we investigate enhancing/increasing multi-source content delivery efficiency in the context of Content-Centric Networking (CCN) with the usage of fountain codes. In particular, we examine whether the combination of fountain coding with the in-network caching capabilities of CCN can further improve performance. We also present an enhancement of CCN's Interest forwarding mechanism that aims at minimizing duplicate transmissions that may occur in a multi-source transmission scenario, where all available content providers and caches with matching (cached) content transmit data packets simultaneously. Our simulations indicate that the use of fountain coding in CCN is a valid approach that further increases network performance compared to traditional schemes.

1. INTRODUCTION

Content-Centric Networking (CCN) [13], also referred to as Information-Centric Networking (ICN) [23], is emerging as one of the principle future networking paradigms. The vast majority of Internet activities are related to content retrieval and the location of the requested content is becoming less important. In recent years, several ICN architectures that enable content access and delivery based on network location-independent names, instead of endpoint addresses, have been proposed. Despite their differences, all ICN architectures view in-network caching as an intrinsic building block. Each packet is uniquely identified and authenticated without being associated to a specific host; cache-enabled nodes in the network cache passing-by data packets. In principle, network caches can serve future matching interests (anycast), therefore data requests do not have to get forwarded to the content origin/server.

According to this line of thought, research has recently focused on the optimization of the in-network caching system in order to improve efficiency, reduce delivery delay and inter-domain traffic. The *cache-everything-everywhere* scheme [13] has been shown to be suboptimal and, as a result of that, a plethora of probabilistic caching approaches have been recently proposed, in which routers lying on the delivery path of an item decide, based on a probability, whether or not to cache passing by packets [12]. In-network, packet-level caching in CCN has been extensively studied from different perspectives, such as content popularity estimation (*e.g.,* [4]), criteria for determining the probability of performing local caching (*e.g.,* [5]) and techniques to reduce caching redundancy (*e.g.,* [19]). The work presented in this paper is orthogonal to all those previous caching work, which mostly look at how and where to cache content.

Fountain coding is an information-theoretic approach for reliably transmitting data from one or more senders to one or more receivers; *i.e.,* fountain codes can be used for data multicasting and multi-sourcing [3, 18]. The fountain coding approach was originally designed for point-to-multipoint transmissions of large files over lossy channels [14, 15, 20], but the use of coding in content distribution networks (*e.g.,* [6, 8, 11]), content retrieval in sensor networks (*e.g.,* [7, 9]) and ICN (*e.g.,* [17]) has already demonstrated significant benefits.

In this paper, we explore the usage and potential benefits of fountain coding for disseminating content in the CCN/NDN [13] architecture and investigate how in-network caching and the reuse of encoding symbols can further enhance performance in such a multipoint-to-multipoint environment. Such enhancement could substantially benefit use cases such as content dissemination to wireless/mobile users, especially in the presence of content replication or even peer-to-peer communication *i.e.,* in the presence of multiple content sources, while keeping inter-source and/or sender-receiver coordination overheads to a minimum.

In the considered NDN architecture, whenever an *Interest* packet arrives at some face of a router, the router searches for matching content in the *Content Store (CS)* and, if such content is found, it responds

with the respective *Data* packet. The router sends the Data packet to the face the Interest arrived at. The Interest is satisfied and therefore discarded (per packet Interest). If the router does not find any matching content in the CS and there is an exact-match *Pending Interest Table (PIT)* entry, the interest's arrival face is added to the PIT entry's *Requesting Faces* list and the Interest packet is discarded. Otherwise, if there is a matching *Forwarding Information Base (FIB)* entry, the Interest packet is sent upstream towards the content provider. In particular, the arrival face is removed from the face list of the matching FIB entry, and if the resulting list is not empty, the packet is sent out to one or more of the remaining faces (in case of multiple content origins/replicas and based on some *Forwarding Strategy* [13]); a new PIT entry that records the Interest and its arrival face is created.

In NDN a router may forward an incoming Interest towards all router interfaces that point to a provider of the requested content (FIB entries [13]). This could lead to multiple copies of the same packet reducing the effectiveness of caching and multi-sourcing. To minimize this overhead, we further enhance the original Interest forward mechanism with a new component named *"Retrieved Information Base" (RIB)*, so that routers that belong on the same path do not send the same cached packets towards the interested user. Finally, we compare the performance of fountain coding in NDN against the traditional NDN in-network caching scheme and other probabilistic caching schemes.

The rest of the paper is organized as follows. In Section 2, we present through a simple example the functionality of the fountain coding and its advantages that led us to consider exploiting it in CCN, whereas in Section 3 we describe the application of fountain codes in CCN/NDN architecture. In Section 4, we evaluate the performance of caching and reusing encoded packets in a multi-source environment, as well as the performance benefits of the newly introduced RIB component. Finally, Section 5 concludes the paper and outlines directions for future research.

2. BACKGROUND OF FOUNTAIN CODING

In fountain coding, a piece of content (content item) is initially fragmented to equally sized fragments (F_1 to F_5 in Figure 1). Using these fragments, a sender/content provider can then construct and send a very large number of encoding symbols; each symbol has the same size as any fragment. Encoding symbols can be constructed on the fly, as required, or pre-constructed and stored prior to any request for the content item. In total, the number of encoding symbols required to decode the requested content is slightly larger than the number of its fragments. Receivers can recover the requested item from any such set of symbols therefore symbol losses and reordering are not important [14].

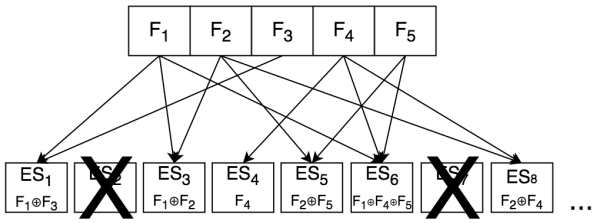

Figure 1: Fountain Coding Example

Encoding. To construct an encoding symbol, a sender first calculates its *degree*, which is sampled from a *degree distribution* (*e.g.,* the *Robust Soliton* distribution in [14]). The degree specifies the number of content fragments that will be encoded (XORed) in each symbol. The properties of the distribution are crucial for efficient encoding and decoding as well as for minimising network overhead [14, 15, 20]. After calculating the degree, the sender selects uniformly at random *degree* fragments as *neighbours* of the encoding symbol (an edge is connecting each encoding symbol with all its neighbours in the bipartite graph shown in Figure 1). The neighbours are XORed together to form an *Encoding Symbol (ES)*. For example, ES_1 has degree *2* and is the result of XORing fragment F_1 with F_3.

Decoding. In its simplest form, decoding is based on belief propagation [14]. A receiver utilises a symbol of degree one (*i.e.,* a decoded symbol - a fragment of the original data) to partially or fully decode other symbols by XORing them with it (*e.g.,* ES_4 in Figure 1). Decoding a fragment may result to further decoding a large number (or even all) of previously received and yet undecoded symbols.[1]

For the example illustrated in Figure 1, let us assume that the sender encodes and sends symbols ES_1 to ES_8 but the receiver receives them in the following order:

ES_1, ES_6, ES_3, ES_4, ES_5, ES_8

Note that encoding symbols ES_2 and ES_7 are erased (lost while being transferred), while some arrive "out-of-order" (quoted since there is no concept of ordering in fountain coding) as depicted in Figure 1. Upon receiving symbol ES_1, the receiver locally stores it since two original fragments (F_1 and F_3) are XORed in it and no decoding can be performed. It also stores symbols ES_6 and ES_3 for the same reason. The next received symbol, ES_4, has degree one and only contains F_4 which is decoded. This symbol is also XORed with all previously stored symbols that contain fragment F_4 (in this case only with symbol ES_6). This action effectively removes the edge between nodes F_4 and ES_6 in the bipartite graph shown in Figure 1. ES_6 now contains fragments F_1 and F_5; fragment F_4 has been *XORed out* of symbol ES_6. The next received symbol, ES_5, with degree

[1]Both encoding and decoding operations are associated with processing (and the corresponding delay) overheads. We plan to focus on these overheads in our future work.

2, contains fragments F_2 and F_5, none of which is yet decoded; therefore it is stored until it can be decoded. The last symbol, ES_8 contains fragments F_2 and F_4. F_4 has been previously decoded and it can be used to decode F_2 by XORing it out of ES_8 (removing the edge between nodes F_4 and ES_8 in Figure 1). In turn, this creates a chain decoding. The newly decoded fragment F_2 decodes F_1 and F_5 from the previously received symbols ES_3 and ES_5, respectively. Finally, F_1 decodes F_3 from ES_1. At this point the original piece of content is decoded.

Fountain coding is an ideal approach for supporting the following types of data transport, all of which are compatible with CCN/NDN:

Multicast. Fountain coding has been a major component of reliable multicasting proposals, such as [3], where clients acquire bulk data at times of their choosing without requiring feedback channels. Congestion control can moreover be implemented in a layered, receiver-driven fashion, where a client reacts to network congestion by registering to and unregistering from different layers (multicast groups) [16].

Multi-Source. Regardless of the number of receivers, multiple senders can contribute to the production of encoding symbols. As long as all sources of the data produce encoding symbols based on the procedure describe above and these symbols are not identical, then all symbols equally contribute to the decoding of the data at the receiver's side. Note that ensuring symbol uniqueness is a matter of randomising the creation of seeds that are used for the calculation of the degree and the neighbours set for each symbol.

Multi-Path and Multi-Home. Encoding symbols may follow different paths in the network without affecting decoding or its efficiency. Symbols may also arrive through different network interfaces. The receiver is oblivious to that, only requiring a specific number of encoding symbols to decode the original data.

3. CCN WITH FOUNTAIN CODES

In this section we describe the application of fountain codes in the CCN/NDN architecture. We also present an enhancement of the Interest forwarding mechanism that alleviates the overhead of potential duplicate transmissions of Data packets in multipoint-to-multipoint communication scenarios.

With fountain coding, each symbol is sent along with some information that the receiver uses to extract its degree and neighbours (*e.g.,* the seed that was used to (pseudo) randomly generate these values). Such information can be part of the name of each CCN/NDN data packet. In the NDN architecture [13] data exchange is clocked by Interest Requests which are sent by receivers in the network. Each packet of the content item is requested separately and the network is responsible for forwarding interest requests towards a source of the data. Sending an Interest for each data packet wastes uplink bandwidth and burdens routers with the continuous maintenance of large amounts of state [22]. Furthermore, if an Interest is lost, the corresponding data packet will not be forwarded, reducing the perceived quality of service. In this paper we follow an approach with Persistent Interests (PIs) similar to [22], where routers store PIs (in the PIT) for a period of time; PIs are not deleted after a matching data packet is forwarded. Instead, they remain in the PIT until users explicitly unsubscribe (*i.e.,* when the requested content item has been retrieved) or their lifetime expires. We believe that the usage of PIs and fountain coding fits better in a multi-source environment, as also shown in our performance analysis.

Naming encoding symbols. Interests name a piece of content as a whole (*e.g., /bbc/item1*). A data packet that satisfies such an interest is named like */bbc/item1/seed*, where *seed* is the seed that was used for pseudo-randomly generating the degree and neighbours for the encoding symbol carried in this data packet. On the receiver's side, degree and neighbours are calculated based on the *seed* provided in the name.

To enable caching, one could use the following name convention for Interest packets: */bbc/item1/RIB*, where *RIB*, i.e., *Retrieved Information Base (RIB)*, is a new name component that represents the seeds of all encoding symbols that are cached by routers (at the time the Interest was processed) on the path that has been followed by the Interest so far. Essentially, with the RIB component, each router can know which of its currently cached symbols have not been already replayed to the receiver(s) by other routers in the path. RIB is updated at each hop along the path followed by the Interest and is used to minimize the overhead of duplicate transmissions.

A client initially sends an Interest with an empty RIB. Each router along a path towards a content origin includes in the RIB the seeds of the matching cached encoded symbols that will be sent back to the client in response to the received Interest. Finally, the content origin, upon receiving an Interest, sends to the client encoded symbols that are not included in the RIB until either the PI expires or the user notifies the server that the content item has been retrieved (and fully decoded).

A realization of the RIB could be through the usage of *Bloom filters* [1]. It is well known that the usage of Bloom filters may yield *false positives*. In our case this means that a router might erroneously assume that cached packets have already been transmitted to the requesting user. It is important to highlight that false positives do not result in the delivery of incorrect data when fountain coding is used since other encoded symbols can be used to decode a content item. On the

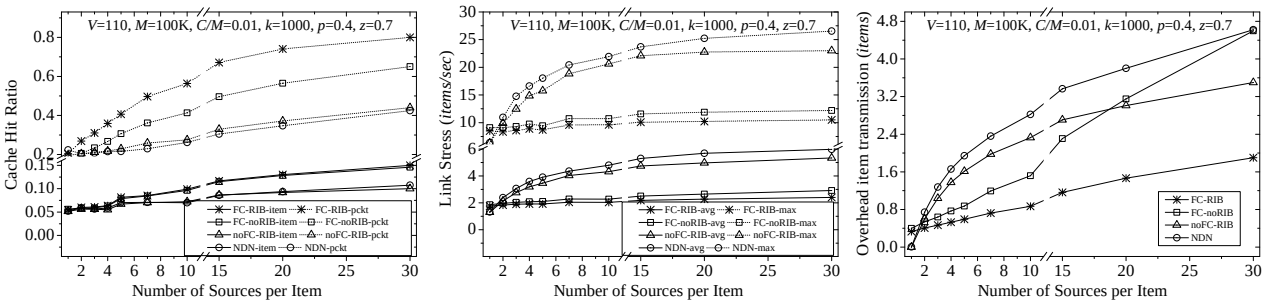

Figure 2: Effect of the number of different content sources per item on the performance of the considered caching schemes.

downside, a false positive might result in some extra traffic, since other encoded packets should be transmitted from another router further along the path or from the content origin itself. Nevertheless, the probability of a false positive depends on the number of bits used and the number of hash functions [2]. For example, using four bits per seed we can create filters for ten thousand encoded packets using less than a five KB packet and yielding an accuracy in the area of 85% (less than 15% of false positives). Further accuracy can be achieved using more bits and larger packets or compressed Bloom filters. Due to lack of space, in our evaluation we assume that false positives are equal to zero; we leave the inclusion of false positives for future investigation.

In-network caching. A CCN router can replay all (or a number of cached encoding symbols) from its cache. In most cases, a cache will have to check whether any of the currently cached symbols for the requested content item has not been previously received by the receiver. The *RIB* component in the Interest packet can be used for that purpose.

In this paper, we assume that each intermediate router along the path from the user to a content provider only sends cached encoding symbols. Only content providers send newly constructed encoding symbols. Another alternative would be to allow network caches to decode pieces of content using passing-by symbols, instead of merely caching them. This would, in turn, allow them to create new encoding symbols, just as original content providers do, minimising the probability of disseminating duplicate encoding symbols. This alternative approach is left for future work. Also, since the cache-everything-everywhere approach of CCN/NDN has been shown to be suboptimal, we assume a simple opportunistic caching mechanism where each router caches an encoding symbol with probability p ($p = 1$ in NDN), regardless of being cached or not elsewhere along the path. More sophisticated approaches where the degree distribution of the coding process is taken into consideration when deciding which symbol to cache, are also left for future investigation.

4. PERFORMANCE EVALUATION

We implemented the proposed fountain coding approach in CCN in a Matlab-based simulator and we evaluate the performance of various caching mechanisms based on the Interoute network topology as provided in the Zoo topology dataset[2]. The Interoute network has $V = 110$ nodes, and essentially spans over the European continent. In the considered scenario, each node is associated with a cache and we assume that, by default, all caches have the same capacity C, defined as a percentage of the content catalogue size M.

In [21], the authors indicate that in their VoD dataset the size of the catalogue of watched videos amounts to $500K$ unique items. In our experimentation, we consider a list of $M = 100K$ content items, a number that preserves a good approximation of a realistic content catalogue while allowing our simulations to run in realistic time-scales. Additionally, and without loss of generality we assume that each item is fragmented into $k = 1000$ equally sized fragments. We also assume that each content item is served by S content origins, that represent servers which are arbitrarily attached to S (different for each item) edge nodes.

To generate user demand, we assumed that item popularity is given by a Zipf law distribution of exponent $z = 0.7$ at each network node/router, and we assumed that users attached to each router are requesting content items with a rate of one request per second. Thus, the request rate for each item at each node varies from 0-1 req/sec depending on item's popularity and ranking. Generally, the popularity of each content item may differ from place to place, a phenomenon that is referred to as locality of interest (*spatial skew* in [10]). In our model, we assume that the request popularity exponent has the same value z at each router; this captures the global popularity of items. In each router the ranking/order of the items within the Zipf distribution is different; this captures the different locality of interest.

We have simulated four different caching schemes. More specifically, we evaluated caching schemes when using fountain coding with and without the RIB compo-

[2]http://www.topology-zoo.org/dataset.html

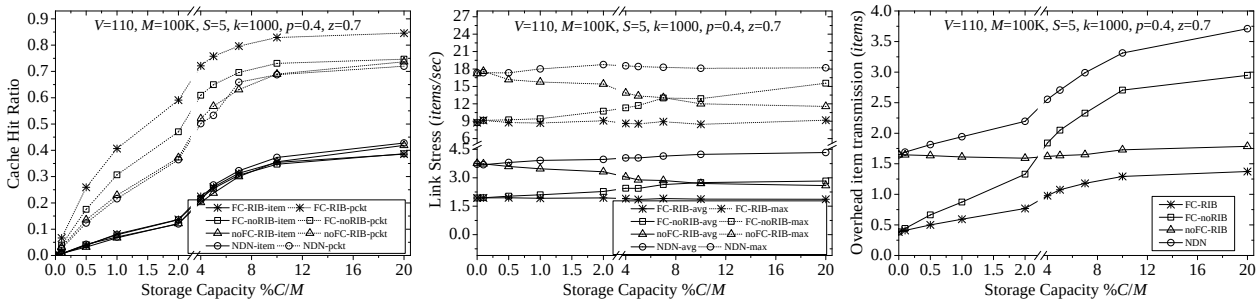

Figure 3: Effect of the caching capacity of each router on the performance of the considered caching schemes.

nent (*FC-RIB* and *FC-noRIB* accordingly), as well as the standard NDN caching mechanism with and without the RIB component (*noFC-RIB* and *NDN* accordingly). In the fountain coding-based schemes as well as in the noFC-RIB scheme, we assumed that each router caches a passing-by packet with a probability equal to $p = 0.4$ ($p = 1$ in the NDN scheme).

Our evaluation is based on the following metrics:

- *Cache Hit Ratio*: We present both the item level as well as the packet level hit ratio. The item level hit ratio is the ratio of the content requests that have found the requested item (all its fragments/packets) cached within the network, over the total number of issued requests. The packet level hit ratio is the ratio of the packets (all the retrieved packets encoded or not) found in the network (and not at the content origins) and used for the decoding of an item for the fountain coding schemes and for the retrieval of an item for the standard NDN schemes (not duplicate found packets).
- *Average Link Stress* (in *items/sec*): the mean number of delivered content items that traverse each link of the network.
- *Maximum Link Stress* (in *items/sec*): the maximum number of delivered content items that traverse the most constrained/congested link of the network. This metric along with the Average Link Stress metric is indicative of the load balancing capabilities of each examined scheme.
- *Traffic Overhead* (in *items*): the average number of extra items that a requesting user receives from the network in order to receive the requested content. This is the amount of content that the RIB component (wherever used) and the duplicate dropping mechanisms of the NDN architecture did not manage to save due to the multi-source environment and the simultaneous request of every available content origin.

Impact of the number of content sources. In Figure 2 we depict the impact of the number of different content sources S in the examined schemes. Generally, we observe that the usage of fountain coding significantly outperforms the traditional schemes when the number of sources are more than one. Only when there is a single content origin the traditional NDN caching mechanism outperforms the fountain coding-based schemes, given the overhead (in terms of required encoding symbols) that is associated with the decoding of the original data. The fountain coding schemes are not affected by the number of sources regarding the link stress metrics and they perform up to 3 times better regarding the packet level hit ratio (2 times better regarding the item level hit ratio) compared to the traditional NDN caching mechanism. Also, the usage of the RIB retains the network overhead at a tolerable level and its usage, combined with the simple probabilistic caching method, performs up to 80% better compared to the cache-everything-everywhere NDN scheme. Finally, the proposed RIB mechanism (even without being combined with fountain coding) is useful for the minimization of the overhead in CCN architectures and definitely worth further exploration.

Impact of the cache size. In Figure 3 we illustrate the impact of the cache capacity, expressed as the fraction of the items' population that can be stored in the cache of a router for a given number of servers ($S = 5$) per content item. As previously, the schemes that incorporate the fountain coding perform significantly better compared to the other ones. Note that for low cache capacities ($\leq 2\%$ of the total information), we observe that the packet level hit ratio is twice as high as in the traditional NDN scheme. This means that in real world scenarios (small cache size at each router), even if the item level hit ratio is small, more than 50% of the delivered packets are found in the network, which, in turn, means that combining fountain coding with our proposed RIB approach could be proven very useful for future ICN applications. Finally, as in the previous experiment, the usage of the RIB component can save up to 70% of the duplicate transmissions even for very low caching capacities.

5. CONCLUSIONS

In this paper we examined the usage of fountain cod-

ing in Content-Centric Networks. Particularly, we investigated the performance of the NDN architecture and its in-network caching capability when combined with fountain coding. We also proposed an enhancement to the NDN Interest forwarding mechanism to minimize the duplicate transmissions in a multi-source environment, where all possible content origins are requested simultaneously. Our simulation-based analysis shows that the usage of fountain coding in CCN is a valid tool for the retrieval of cached content since it significantly outperforms traditional schemes.

Apart from the future work hints given throughout the paper, this work can also be extended to accommodate the retrieval of cached content in disruptive scenarios, where the content origin is not always present due to its own mobility or to network node/link failures and the transmission of a packet between two nodes is prone to errors.

6. REFERENCES

[1] B. H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 1970.

[2] J. W. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In *ACM SIGCOMM*, 2002.

[3] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM*, 1998.

[4] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino. Modeling data transfer in content-centric networking. In *ITC*, 2011.

[5] W. K. Chai, D. He, I. Psaras, and G. Pavlou. Cache "less for more" in information-centric networks (extended version). *Computer Communications*, 2013.

[6] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *Information Theory, IEEE Transactions on*, 2010.

[7] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Ubiquitous Access to Distributed Data in Large-scale Sensor Networks Through Decentralized Erasure Codes. In *IPSN*, 2005.

[8] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Decentralized erasure codes for distributed networked storage. *IEEE/ACM Transactions on Networking (TON)*, 2006.

[9] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Distributed fountain codes for networked storage. In *IEEE ICASSP*, 2006.

[10] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less Pain, Most of the Gain: Incrementally Deployable ICN. In *ACM SIGCOMM*, 2013.

[11] C. Gkantsidis and P. R. Rodriguez. Network coding for large scale content distribution. In *IEEE INFOCOM*, 2005.

[12] A. Ioannou and S. Weber. Towards on-path caching alternatives in Information-Centric Networks. In *IEEE Conference on Local Computer Networks, LCN*, 2014.

[13] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *ACM CoNEXT*, 2009.

[14] M. Luby. LT Codes. In *Symposium on Foundations of Computer Science*, 2002.

[15] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer. RFC 5053 - Raptor Forward Error Correction Scheme for Object Delivery. Technical report.

[16] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. *SIGCOMM Comput. Commun. Rev.*, 1996.

[17] M.-J. Montpetit, C. Westphal, and D. Trossen. Network Coding Meets Information-centric Networking: An Architectural Case for Information Dispersion Through Native Network Coding. In *ACM NoM*, 2012.

[18] G. Parisis, T. Moncaster, A. Madhavapeddy, and J. Crowcroft. Trevi: Watering Down Storage Hotspots with Cool Fountain Codes. In *ACM HotNets*, 2013.

[19] I. Psaras, W. K. Chai, and G. Pavlou. In-Network Cache Management and Resource Allocation for Information-Centric Networks. *IEEE TPDS*, 2014.

[20] A. Shokrollahi. Raptor codes. *Information Theory, IEEE Trans. on*, 2006.

[21] Y. Sun, S. K. Fayaz, Y. Guo, V. Sekar, Y. Jin, M. A. Kaafar, and S. Uhlig. Trace-Driven Analysis of ICN Caching Algorithms on Video-on-Demand Workloads. In *ACM CoNEXT*, 2014.

[22] C. Tsilopoulos and G. Xylomenos. Supporting Diverse Traffic Types in Information Centric Networks. In *ACM SIGCOMM Workshop on ICN*, 2011.

[23] G. Xylomenos, C. N. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos. A survey of information-centric networking research. *Communications Surveys & Tutorials, IEEE*, 2014.