ICN-RE: Redundancy Elimination for Information-Centric Networking

Diego Perino
Bell Labs, Alcatel-Lucent, FR
diego.perino@alcatel-lucent.com

Matteo Varvello
Bell Labs, Alcatel-Lucent, USA
matteo.varvello@alcatel-lucent.com

Krishna P. N. Puttaswamy
Bell Labs, Alcatel-Lucent, USA
krishna.puttaswamy_naga@alcatel-lucent.com

ABSTRACT

This paper bridges Information-Centric Networking (ICN), a novel form of networking centered around information or content, and Redundancy Elimination (RE), a popular technique widely used to identify content with similar bytestream. The result is ICN-RE, the first ICN design that supports redundancy elimination. We show by means of numerical evaluations that ICN-RE improves bandwidth efficiency by 15-40% compared to vanilla ICN, and that the state of the art hardware can support ICN-RE at line speeds up to 100Gbps. Also, we discuss several benefits of adopting ICN-RE as a network-wide RE solution when compared to state of the art redundancy elimination designs.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Designs**]: Network communications; B.3.2 [**Design Styles**]: Cache memories, Mass storage

General Terms

Design, Performance

Keywords

ICN, Redundancy Elimination, Caching

1. INTRODUCTION

The research community has recently proposed *Information-Centric Networking* (ICN) [3], a ground breaking networking approach where *content is requested by its name instead of by its host* (Section 2.1). In ICN, the network elements, such as routers, are aware of the data they transfer. This requires a more complex forwarding mechanism [11], but it enables wide-spread caching that has the potential to significantly reduce data access latency, load on the network and energy cost. The effectiveness of ICN ubiquitous caching, however, is currently under debate [10].

To the best of our knowledge, ICN designs cache content purely based on their names: two content items (packets/chunks) with the same name are considered equal and thus only one copy is stored. For example, consider two documents with content names */bell_labs/page1.html* and */bell_labs/page2.html*; let's say that each of these documents is 2,000 bytes long and that the last 1,000 bytes of the two documents are the same, say some copyright information of *bell_labs*. In ICN, each router forwarding these documents caches the last 1,000 bytes of both documents, although only one copy should be sufficient. This happens because the caching works solely based on *content names* and is unaware that some bytes are common across the two documents.

Redundancy Elimination (RE) is a compelling mechanism used to identify and remove redundant bytes transferred in a network (Section 2.2). A trace-driven analysis [4] has shown that protocol independent redundancy elimination mechanisms can remove from 10% to 60% of redundant data in enterprise and university access links. Several vendors (e.g., Riverbed and Cisco) offer boxes that implement RE for enterprises, data centers and ISP links already. In addition, researchers have already shown how to deploy network-wide RE by adding RE as a primitive IP-layer service on network routers [5]. In this paper, we aim to bridge ICN and RE with the goal of improving the effectiveness of caching in ICN.

We design ICN-RE as an extension of NDN [11], today's most popular ICN design (Section 3). ICN-RE does not impose any RE related overhead on the routers [5], or the clients [2], but offloads the overhead on the content server. The content server is responsible for chunking files such that a redundant sequence of bytes is mapped to a single chunk. Thus, a file in ICN-RE consists of a set of chunks, and each chunk may be a part of different files. We associate to each chunk a "fingerprint" derived by hashing the entire content of the chunk. Caches are indexed using the fingerprints, and lookups are performed via exact match. Finally, a server informs clients, upon requesting a document, about the fingerprints in the requested file using a "manifest" file.

We perform a preliminary evaluation of ICN-RE using bandwidth savings and chunk size distribution derived from the work in [4] (Section 4), a large scale trace-driven study of protocol independent redundancy elimination mechanisms. Our evaluation shows two main results: i) ICN-RE generates at least between 15 and 40% bandwidth savings compared to vanilla ICN; ii) state of the art hardware technology supports ICN-RE at line speeds up to 100Gbps.

Towards the end of the paper, we discuss several benefits of adopting our proposed design for network-wide RE (Section 5). Compared to state of the art RE designs, we estimate that ICN-RE trades about 10% of traffic savings for lower complexity, higher resilience to failures, enhanced storage efficiency and user delivery performance. For these reasons, the deployment of ICN-RE as a network-wide RE scheme drives our future work (Section 6).

2. BACKGROUND

We first provide an overview of NDN. Then, we describe RE mechanisms and their past application. Finally, we discuss how ICN-RE departs from previous RE approaches.

2.1 Named Data Networking

In NDN every content item is divided in a sequence of chunks, each uniquely identified by a hierarchical human-readable name. A content name has B *components* delimited by a character, e.g., `/ICN/PAPERS/PaperA.pdf/chunk0`. Content is requested using an *Interest* packet that contains the name of the desired chunk. An Interest packet propagates toward potential data location using longest prefix match. The Interest propagation leaves a trail of "bread crumbs" that a matching chunk follows to reach back the original requester(s). Interests for the same chunk are aggregated naturally realizing multicast. Content routers cache chunk for later transmission using content names as an indexing key.

Three main components form a content router: *Forwarding Information Base (FIB)*, *Content Store (CS)* and *Pending Interest Table (PIT)*. The FIB is a table that associates prefixes of content names to one or multiple next hop routers. The CS plays the same role of a buffer in an IP router. However, it also stores chunks after they have been forwarded so that they can serve future requests. The PIT keeps track of the chunks that have recently been requested and not yet served, in order to forward chunks on the reverse path towards requesters and to realize interest aggregation.

2.2 Redundancy Elimination (RE)

RE is a popular technique to detect and eliminate similar byte-streams from network traffic irrespective of traffic source or protocol. Several systems have been proposed in the past [2, 5, 12]; despite the differences in the context where they are applied, all these systems require that the two end points respect the following conditions: i) can cache recent packets, ii) have their cache synchronized. If these conditions hold, then RE is achieved by performing the following operations.

Fingerprints computation based on content in the cache: Before a new packet is sent across a link, the sender node computes Rabin fingerprints[1] of the packet and compares them to the fingerprints of the content in the cache. Then, the portions of the new packet around the matching fingerprints are compared to content in the cache in oder to identify the maximal regions of redundant bytes.

Encoding of the packets: Once the redundant content is identified, the packet to be transmitted is encoded by removing the redundant content and replacing them with fingerprints or metadata that point to the content eliminated in the cache. The new encoded packet is then transmitted.

Decoding of the packets: This operation consist of replacing the fingerprints in the encoded packets with the associated content by looking up the fingerprints in the cache.

The RE technique described above is shown to eliminate 10-60% of the overall data in a network, and has been applied in various settings: i) at the exit and entry points of enterprises [12]; ii) as a network-wide RE solution (SmartRE [5]), where every router performs caching, encoding, and decoding; iii) at the end-hosts, where only server and clients are modified without any network changes (EndRE [2]).

ICN-RE departs from these past approaches. A content router, the equivalent of an interior and ingress router in prior solutions, does not perform any encoding or decoding. This is possible because ICN-RE offloads encoding to the source. This is similar to EndRE [2], but ICN-RE is even simpler: clients do not need to cache chunks or perform resource intensive decoding operations. The clients simply combine the various chunks they receive from the network, just the way it is already done in NDN. As a result, in ICN-RE the RE intelligence only needs to be incorporated by modifying the source without changing the routers or to the clients.

[1]The Rabin fingerprinting scheme is a method for implementing public key fingerprints using polynomials over a finite field.

3. ICN-RE

ICN-RE is designed as an extension of NDN even though it could be built on other ICN designs. Our motivations are two fold: i) NDN's hierarchical naming scheme and pull-based approach enable us to easily integrate RE mechanisms; ii) NDN offers a complete prototype that we plan to use in the future to extend this work.

Next we describe the design of ICN-RE, the first ICN design that supports RE. After a brief design rationale, we overview the mechanism we use for redundancy-elimination. Then, we detail the features that we modify from NDN, namely naming, and caching. Remaining NDN mechanisms, such as Interest and chunk forwarding, are unchanged compared to the description in Section 2.1.

3.1 Design Rationale

The key driving force behind the design of ICN-RE is to keep the changes to the existing ICN as minimal as possible. As in, we would like to add little to no additional tasks on the routers and the clients while achieving efficient redundancy elimination. In [4], Anand et al. show that *intra-server* redundancy is the main contributor to the overall redundancy in traffic, whereas the contributions from *inter-server* RE is quite limited. Based on this observation and similarly to [2], our key design decision is to offload the task of redundancy identification and chunking of content to the source.

The source has complete knowledge of the content it serves; thus, it can analyze the content, identify the redundant data and split files into chunks to maximize redundancy elimination. Using this approach, with only a minor change to the way content is named in ICN, we can incorporate RE into ICN. We do not require any changes to the clients. This leads to a significantly simpler design compared to prior art.

3.2 Redundancy Identification and Chunking

We now describe how we integrate some of the techniques from prior art into NDN to enable the source to identify the redundant data across different files it serves, and generate chunks for every document. This operation is a low-overhead task that each source has to do once on its content library, unless the content changes significantly; in this case, this analysis should be redone.

Content Analysis. The goal of the content analysis is to identify as much redundancy as possible across the entire content at the source. To accomplish this, we can reuse any of the fingerprinting techniques proposed in the prior art [5, 2, 12]. The chunking process proceeds as follows: for each file, of size L, a window of size w is moved from the beginning of the file to the end of the file. This window w represents the minimum size of the redundant string (or contiguous sequence of bytes) we would like to identify, and it is usually a number between 32 and 64. Each of the total $L - w + 1$ strings are used to compute a Rabin fingerprint (or any of the other fingerprinting schemes in [2]) generating $L-w+1$ fingerprints. A small fraction of these fingerprints are selected based on their value (the selection criteria varies based on the fingerprinting scheme used). Let's call the first byte in the chosen candidate string (that leads to the chosen fingerprint) a *marker*, and the data between two markers a *chunk*. Each chunk is hashed to obtain a *chunk hash*, also called the *chunk fingerprint*. Variations to this scheme can also be employed by the source to further improve RE efficiency; for example, the chunk expansion used in [5, 2, 12]. At the end of the content analysis, the source knows the chunk composition of any given file, *i.e.,* the list of fingerprints identifying the chunks.

Encoding. In ICN-RE, there is no heavy-weight encoding operation as in the prior RE schemes (Section 2.2). The equivalent of this operation is the computation of the *manifest* file. The manifest contains up-to-date information about the fingerprints that describe a content, and it is transferred from the source to a client before content delivery.

Decoding. In ICN-RE, again, we do not need the heavy-weight decoding operation used in the prior RE schemes. The equivalent of this operation is the aggregation of the content chunks performed at the receiver. Such aggregation is already an integral part of NDN, and hence there are no additional changes necessary at the receiver.

3.3 Naming

ICN-RE uses a hierarchical naming scheme as in the original NDN. However, we add to the naming scheme the information about the chunk fingerprint. In an *Interest* packet, the fingerprint is accommodated at the end of the naming tree or can replace chunking information. For example, in order to request *fingerprint1* in a content with name */bell_labs/page1.html*, the client would issue an Interest addressed to */bell_labs/page1.html/chunk1/fingerprint1* or */bell_labs/page1.html/fingerprint1*. Differently from NDN, the fingerprint only and not the full content name is used to address a chunk. This is possible as: i) ICN-RE uses only the fingerprint, and not the whole content name, to index the cache; ii) content name is not used for forwarding chunks in the reverse request path, as NDN uses bread crumbs routing: the PIT can also be indexed using fingerprints and not the whole content names. This last design choice allows to enhance the efficiency of the native multicast feature of NDN: requests for chunks with same fingerprint but belonging to different documents can also be aggregated. The presence of the fingerprint accounts for additional 16 Bytes[2] in the Interest packet; however, stripping off the $B - 1$ components of the content name from each chunk generates a big save as content names have unbounded length, and can account for up to 200 bytes [1].

Remark that the original NDN naming scheme includes a cryptographic digest component for every chunk already. This component is derived by using the SHA-256 hash function over the entire encoded chunk, including the data payload of the chunk, the content name, signature and other chunk fields. The digest component is not explicitly included in chunks, as it can be easily derived from the chunks themselves (*i.e.,* computing the SHA-256 values), and can be optionally included in Interest packets as last component of the requested item. Differently, the ICN-RE fingerprint is computed on the data payload of the chunk only, so that chunks belonging to different documents (e.g., with different content names) will generate the same fingerprint.

Assuming the client knows the content name that uniquely identifies the content item she aims to retrieve, we need a mechanism to discover the fingerprints associated with the content chunks. We do this by means of a manifest, as described before. The manifest is treated as regular ICN data: it is requested via an Interest packet and can be cached for future reuse. Upon receipt of the manifest, the client learns the chunks composing the file, and then iteratively issues requests for each chunk. In order to ensure consistency between the latest manifest generated at the source and cached copies, a timer is associated to the manifest after which it becomes stale and is removed from the caches. The retrieval of manifest file before the beginning of the actual download introduces an additional start-up delay, that can be non-negligible in case of large files (*i.e.,* large manifest). This drawback can be mitigated by splitting the manifest in multiple small documents (potentially single chunk documents), that are downloaded sequentially when needed. In this case, the actual content retrieval can start as soon as the first part of the manifest is retrieved, causing negligible start-up latency, while the rest of the manifest can be retrieved during the content download.

Finally, chunk name authentication and data integrity should be guaranteed. In NDN, every chunk is publicly authenticatable: this is realized by means of standard public key signatures and Merkle hash trees [11]. This approach does not suite ICN-RE where a chunk can belong to different files, and a fingerprint is associated to multiple content names. Baugher et. al [8] propose to separate data integrity from name authentication. Data integrity is guaranteed by using hash-name to address data chunks: a node can thus easily verify whether a chunk has been altered while traveling on the network. Name authentication is realized by means of a catalog that associates real world names to hash names, the association being optionally signed by a real world entity. The catalog can be safely exchanged by means of different in band and out of band methodologies [8]. Such approach better suites ICN-RE. We plan to further investigate ICN-RE security issues as future work.

3.4 Caching

Unlike NDN, ICN-RE indexes data in the cache based on the content rather than the name of the content. Thus, the fingerprint and not the content name is used as a key to store and lookup chunks in the CS. As a result, whenever two chunks have the same fingerprint, *i.e.,* they are redundant copies of each other, they both generate a cache hit despite the presence of a single cache entry.

We implement the Content Store (CS) by using *HC-basic* algorithm [7]. HC-basic treats the packet store, *i.e.,* the part of the CS where chunks are stored, as a fixed size non-chained hash-table in slow memory. The chunk, along with the fingerprint, is inserted at the location corresponding to fingerprint value modulo the number of *bins*, *i.e.,* the addressable elements of the hash-table. In addition to the original HC-basic design, we have an index table in the fast memory of the router, which is addressed similarly but stores the first H-bit of the fingerprint for each chunk[3]. The index table can generate false positives with probability $1-(1-2^{-H})^N$ where N denotes the number of entries in the packet store.

A lookup in the CS works as follows. First, we extract the fingerprint from an Interest packet and perform a lookup in the index table using the first H-bits of the fingerprint. If a miss occurs in the index table, the Interest is passed to the FIB for forwarding towards potential data sources. Otherwise, the corresponding entry in the packet store is read. Such a read gives as a result the pair <fingerprint,chunk>; we compare the fingerprint read from the packet store with the requested fingerprint. If there is a match, the corresponding chunk is returned downstream; otherwise, we have detected a false positive and the request is passed to the FIB.

Differently from traditional RE designs, ICN-RE does not require synchronized caches installed at source and clients. Chunks can be retrieved from any node in the network, and every cache independently runs a simple replacement policy, e.g., FIFO, random.

[2]When MD5 is used for generating the fingerprint; note that our design is not tied to a specific hash function and other functions can be used.

[3]The index table and the packet store can also be considered as an N-way set-associative hash table to reduce collisions in hash bins, or arrays addressed using a substring of the fingerprint [6].

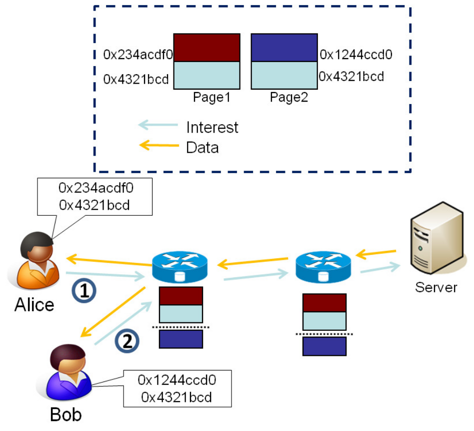


Figure 1: Example of ICN-RE.

3.5 Example

Figure 1 graphically shows the functioning of ICN-RE when two users, Alice and Bob, request two partially redundant documents, page1 and page2. Page1 and page2, with content names */bell_labs/page1.html* and */bell_labs/page2.html*, are composed by two 2,000 bytes chunks with fingerprints 0x234acdf0-0x4321bcd and 0x1244ccd0-0x4321bcd, respectively. For example, each document has the same copyright form which hashes to 0x4321bcd. At t=0, no content is cached in the ICN network.

Alice sends a request for */bell_labs/page1.html* that the ICN network routes to the */bell_labs/* server. As a response, the server transmits the manifest file that contains the fingerprints of the two chunks composing the requested content. Alice uses the information from the manifest to issue Interest packets for chunk */bell_labs/page1.html/0x234acdf0* and for chunk */bell_labs/page1.html/0x4321bcd*. Each Interest is routed to the server by mean of LPM as each in-path cache is currently empty. The server replies to the Interests with chunks containing the actual chunks; both chunks are cached along the path, *i.e.,* 2 chunks and 2,000 bytes.

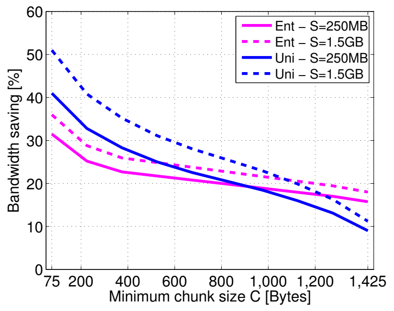
Bob sends a request for */bell_labs/page2.html* and the same procedure described above is performed. Accordingly, Bob sends two Interests, */bell_labs/page2.html/0x1244ccd0* and */bell_labs/page2.html/0x4321bcd*. As above, the first Interest packet propagates to the server as it is unavailable along the path. Also, each router caches the chunk along the path, *i.e.,* 1 chunk of 1,000 bytes. However, the second Interest packet can be served by the first ICN router by re-using the chunk previously cached from *page1*. Note that classic ICN would not detect the presence of a redundant chunk causing wasted space in the cache, and higher network cost.

This very simple example shows all major benefits of ICN-RE: i) each router only stores 3,000 bytes versus 4,000 bytes in NDN, ii) the last chunk incurs shorter delay and lower network cost as it is served from one hop rather than traversing the whole network.

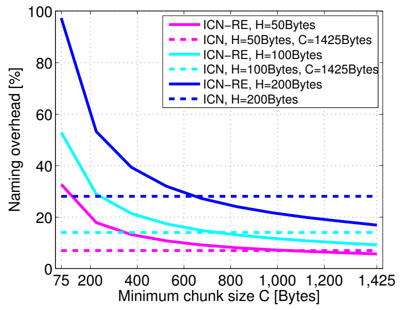
4. EVALUATION

This Section preliminarily evaluates ICN-RE. Our goal is two fold: i) quantify bandwidth savings that ICN-RE enables compared to vanilla ICN, and ii) evaluate suitability of software and hardware technologies for the support of ICN-RE.

We proceed as follows. In [4], the authors conduct a large scale trace-driven study of protocol independent redundancy elimination mechanisms. From this work, we derive bandwidth savings and chunk size distribution for the traffic generated by *a single source* – as it best matches the scheme we propose – in both a typical enterprise and university network. We assume a minimum chunk size C is allowed, *i.e.,* the source emits chunks whose sizes are distributed as in [4] and chunks smaller than C are merged in a single larger chunk. Then, we use this data to estimate bandwidth savings that ICN-RE produces compared to a vanilla ICN. Finally, from the chunk size distribution we derive the rate of processing operation that a ICN-RE router has to perform; this is used to give specifications on the CS, PIT and FIB deployment.



(a) Bandwidth saving ; $C = [75 : 1425]$Bytes ; $S = [0.25; 1.5]$GBytes ; Ent/Uni traces.



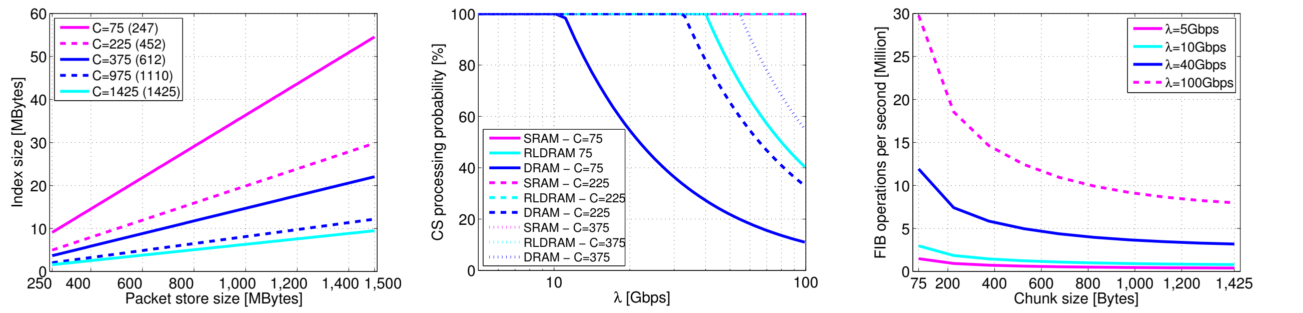
(b) Naming overhead ; $C = [75 : 1425]$Bytes ; Uni trace.

Figure 2: Analysis of potential bandwidth saving and naming overhead.

4.1 Results

Figure 2(a) shows the bandwidth savings as a function of the minimum chunk size C for the enterprise (Ent) and university (Uni) traces, considering cache size S ranging from 0.25 to 1.5GBytes as in [4]. Figure 2(a) shows that for both traces the peak of the bandwidth savings results with the minimum chunk size (C=75Bytes), e.g., 40-50% of the overall traffic at the university and 30-35% at the enterprise. Increasing the value of C has two effects. First, the bandwidth savings are reduced, e.g., only 10% at the university and 18% at the enterprise when C=1425Bytes; this happens because the efficiency of the RE scheme reduces with large chunk sizes, as the probability to find redundant data decreases. Second, the gap between the bandwidth savings among the two cache sizes decreases, e.g., from 5% (enterprise) to 10% (university) when C=75Bytes to 2% (both traces) when C=1425Bytes; this happens because as the efficiency of the redundancy elimination decreases, the additional space available on the largest cache is not exploited.

Since we had no access to the traces, we could not directly quantify the bandwidth savings that caching can generate in a vanilla ICN. Fortunately, we can derive an estimation from Figure 2(a).



(a) Index size ; $S=[0.25:1.5]$GBytes ; $C=[75;225;375]$Bytes; Uni trace.

(b) CS Processing probability ; λ=[0.1:100]Gpbs ; $C=[75;225;375]$Bytes; SRAM/RLDRAM.

(c) FIB analysis; $C=[75:1425]$Bytes ; $S=0.25$GBytes; Uni trace.

Figure 3: ICN-RE Analysis of index size overhead and processing capacity of the content store.

When C is reduced, the redundancy elimination efficiency decreases: this means that less content is found redundant across chunks, and bandwidth savings mostly arise from caching. It follows that the bandwidth savings obtained when C=1425Bytes are an upper bound of the vanilla ICN scenario. This suggests that ICN-RE generates at least between 15 and 40% additional bandwidth savings compared to vanilla ICN.

ICN-RE generates an additional overhead with respect to vanilla ICN for the distribution of the manifest file. The size of this file depends on the average chunk size, which is in turn related to the minimum chunk size C. Figure 3(a) reports in the legend both C and, in the brackets, the respective average chunk size for the Uni trace. Since we use 16Bytes for the fingerprint of each chunk, the size of the manifest varies between 1% ($C=1,425$Bytes) and 7% ($C=75$Bytes) of the file size. In addition, caching of the manifest file can further reduce the additional overhead.

This overhead can be mitigated by the saving related to ICN-RE naming scheme: vanilla ICN uses the full content name (up to 200Bytes) to address chunks while ICN-RE only uses fingerprints (16Bytes). However in ICN-RE, more Interests/chunks are exchanged for each content retrieval due to the smaller chunk sizes compared to vanilla ICN. Figure 2(b) quantifies the overhead, related to the ICN-RE and vanilla ICN naming schemes, that is generated for the retrieval of a chunk (*i.e.,* Interest/chunk exchange). We assume vanilla ICN uses 1,425Bytes chunks and a content name of size H is included in both Interests and chunks. In ICN-RE content name and fingerprint are used in Interest packets, whereas chunks only contain fingerprints. Results are reported for the Uni trace.

Figure 2(b) shows that for minimum chunk sizes larger some thresholds, *i.e.,* 700, 800 and 1,100Bytes for H=200,100,and 50Bytes respectively, ICN-RE reduces the overhead with respect to vanilla ICN. This indicates that our approach can reduce bandwidth consumption without increasing (or even reducing) the communication overhead. For instance, for H=100Bytes, C=700Bytes, and S=1.5GBytes, ICN-RE saves about 28% more bandwidth than vanilla ICN without increasing the communication overhead. For minimum chunk sizes smaller than the above thresholds, the overhead generated by ICN-RE increases and mitigates the potential bandwidth savings. For example, for H=100Bytes, C=200Bytes, and S=1.5GBytes, bandwidth savings are reduced to about 18% because of the communication overhead (with respect to the potential 30%). Finally, for very small minimum chunk sizes the naming overhead can completely nullify the potential benefits of RE.

We now investigate the hardware requirements of the CS to support ICN-RE. Figure 3(a) shows the memory required to store the index table as a function of the packet store size, considering several values of C. We set H=40 to guarantee a false positive probability of about 10^{-6} for a content store with 6Million entries, e.g., 1.5GBytes with C=75Bytes. Intuitively, as C increases the size of the index table decreases, whereas the size of the index table increases as the packet store size increases. However, even in the worst case scenario, *i.e.,* C=75Bytes and packet store of 1.5GB, the index table has a size of only 50MBytes. Such small amount of memory can easily be stored on off-chip RLDRAM or even SRAM, *i.e.,* the fastest (but smallest) memory on the market.

We now estimate the line speed that the CS can handle according to the employed hardware technology. Figure 3(b) shows the fraction of chunks that can be processed at the CS as the link speed λ varies from few Gbps up to 100Gbps, *i.e.,* next generation line speed. We consider SRAM, RLDRAM and DRAM as memories for the index table and several indicative values for C. The Figure shows that as C increases, e.g., C>=375Bytes, any memory but DRAM can support even the fastest link at 100Gbps. Similarly, when λ<10Gbps, the slowest memory (DRAM) is enough to allow 100% processing probability even for the smallest chunks, C=75Bytes. However, assuming a 40/50Gbps link, SRAM memory should be used if the smallest chunk size is allowed (*i.e.,* maximum bandwidth savings), as the RLDRAM cannot process up to 20% of the Interest. As discussed above, the index table fits on both SRAM and RLDRAM, which makes all the discussed configurations perfectly feasible.

Finally, we evaluate the FIB in ICN-RE assuming Interests are not aggregated in the PIT. On the one hand, the overall higher number of Interest causes additional forwarding operations. On the other hand, the higher cache efficiency reduces the number of packets that need to be forwarded. Figure 3.5 shows the number of FIB operations as a function of C when λ=5,10,40,100Gbps and $S=0.25$GBytes. As before, we consider the values derived for C=1425Bytes as representative for the vanilla ICN. Accordingly, when B=75Bytes, *i.e.,* maximum bandwidth savings in ICN-RE, the number of forwarding operations in ICN-RE triplicates compared to a vanilla ICN, independently of λ. ICN-RE and a vanilla ICN have a comparable amount of forwarding operation when C=1,000Bytes, which still guarantees 10% bandwidth savings compared to vanilla ICN (Figure 2(a)). These results also apply to the PIT, as PIT and FIB perform the same amount of operations when Interests are not aggregated.

5. DISCUSSION

This paper focused on the benefits that RE can generate for ICN. However, we believe that ICN-RE is also an interesting approach for network wide redundancy elimination in today's networks. Next we compare ICN-RE with SmartRE [5] and EndRE [2], the most advanced RE designs proposed so far.

Complexity – We define complexity as the total number of operations associated with a data transfer. In SmartRE, once a connection is set-up each packet transmission consists of the following operations: encoding and fingerprint computation (ingress router), decoding (at each interior router). In EndRE, encoding and fingerprint computation are done on-line at the server, while the clients perform the decoding. These operations are much simpler in ICN-RE: encoding is only done at the source off-line, whereas decoding simply consists of chunk reassembling at the client.

Resilience – Resilience is the capability to obtain data in case of failures such as cache misses. SmartRE and EndRE invest a lot of resources to keep failures minimum, as a failure can cause a complete retransmission of the data. ICN-RE is much more resilient to failure thanks to its pull-based nature: every time an Interest cannot be served by a cache, it is simply propagated upstream so that either another router can serve it or the original source can.

RE Efficiency – We define RE efficiency as the volume of redundant traffic that can be saved. SmartRE implements a complex optimization suite to maximize bandwidth savings, that eliminates both inter and intra-server redundancy. ICN-RE, as EndRE, cannot handle inter-server redundancy (on purpose). Both ICN-RE and EndRE aim to be simple so as to be largely scalable and easily deployable. The drawback is a 10% reduction, compared to SmartRE, in the traffic savings originated by inter-server redundancy [4]. Note that we could extend ICN-RE to better account for inter-server redundancy by taking some input from the network and then tuning the chunking scheme at the server.

Storage Efficiency – In SmartRE and EndRE caches are synchronized, *i.e.,* they store the same data by design. On the contrary, in ICN-RE every cache independently runs its own replacement policy, and cache coordination techniques can be employed to differentiate data stored in the network. The desired data can then be found in any node and transferred to the user. Overall, this saves a large amount of storage that can be used for additional data.

End-User Performance – SmartRE and EndRE cannot improve the user delivery performance, as encoded packets always have to travel from the source to the users. On the contrary, in ICN-RE data can be provided by any cache in the network ensuring lower delay and higher bandwidth. Data is served by the source only when it is not available in any intermediate node.

Deployability – SmartRE and EndRE are solutions designed to work on the end-to-end paradigm, and thus fully compatible with today's Internet. ICN-RE is designed for a content-centric network, not yet available today. However, several technical solutions enable content-centric features in today's IP, e.g., [9].

Extensible Chunking – So far, we have described the chunking at the source with the goal of maximizing the common content identified. However, differently from SmartRE and EndRE, ICN-RE allows varying the set of files considered for chunking, which can lead to different combination of chunks. For instance, the chunking could be optimized to minimize access latency to popular files at the cost of increasing the latency for the non-popular ones. Similarly, chunking can be optimized for paid vs. unpaid content, content popular by time (day vs. night), among others.

6. CONCLUSIONS AND FUTURE WORK

Information-Centric Networking (ICN) is a novel networking paradigm centered around information or content distribution. ICN enables wide-spread caching of data in network elements (including routers) to reduce data access latency, load on the network and energy costs. To the best of our knowledge, existing ICN designs cache content data from several documents by using a key which is specific to each <chunk,document> pair. This leads to caching of a significant amount of duplicate bytes: bytes that are common in different documents are cached multiple times as they map to different keys. This duplication can severely hurt ICN's efficiency due to its widespread caching feature.

In this paper we design ICN-RE, the first content-centric design that suppress redundant content. We perform a preliminary evaluation of ICN-RE by focusing on bandwidth savings and deployability. Our numerical results show that ICN-RE can save between 15 and 40% of bandwidth compared to a vanilla ICN. Also, ICN-RE is lightweight and support line speeds up to 100Gbps.

Our road map for future work consists of: i) prototyping ICN-RE, and performing an extensive evaluation, 2) comparing ICN-RE with existing redundancy elimination designs.

Acknowledgments

This work has been partially funded by the French national research agency (ANR), CONNECT project, under grant number ANR-10-VERS-001.

7. REFERENCES

[1] NDN proposal. Website. `http://www.named-data.org/ndn-proj.pdf`.

[2] B. Aggarwal, A. Akella, A. Anand, A. Balachandran, P. Chitnis, C. Muthukrishnan, R. Ramjee, and G. Varghese. Endre: An end-system redundancy elimination service for enterprises. In *Proc. of NSDI*, 2010.

[3] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A Survey of Information-Centric Networking. In *Dagstuhl ICN Seminar*, 2011.

[4] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee. Redundancy in network traffic: findings and implications. In *Proc. of SIGMETRICS*. ACM, 2009.

[5] A. Anand, V. Sekar, and A. Akella. Smartre: an architecture for coordinated network-wide redundancy elimination. In *Proc. of SIGCOMM*, New York, NY, USA, 2009. ACM.

[6] S. Arianfar, P. Nikander, and J. Ott. On content-centric router design and implications. In *ReARCH'10*, Philadelphia, PA, USA, Dec. 2010.

[7] A. Badam, K. Park, V. S. Pai, and L. L. Peterson. Hashcache: cache storage for the next billion. In *Proc. of NSDI*, Boston, Apr. 2009.

[8] M. Baugher, B. Davie, A. Narayanan, and D. Oran. Self-verifying names for read-only named data. In *IEEE INFOCOM NOMEN Workshop*, pages 274–279, 2012.

[9] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini. Conet: a content centric inter-networking architecture. In *Proc. of Information-centric networking (ICN)*, 2011.

[10] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proc. of HotNets*, New York, NY, USA, 2011. ACM.

[11] V. Jacobson, D. K. Smetters, J. D. Thronton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Network Named Content. In *CoNEXT '09*, Rome, Italy, Dec. 2009.

[12] N. Spring and D. Wetherall. A protocol-independent technique for eliminating redundant network traffic. 2000.