In-Network Caching vs. Redundancy Elimination

Liang Wang[†], Walter Wong[*], Jussi Kangasharju[†‡]

[†]Department of Computer Science, University of Helsinki, Finland
[*]School of Electrical and Computer Engineering, University of Campinas, Brazil
[‡]Helsinki Institute for Information Technology, University of Helsinki, Finland

***Abstract*—Network-level Redundancy Elimination (RE) techniques have been proposed to reduce the amount of traffic in the Internet. and the costs of the WAN access in the Internet. RE middleboxes are usually placed in the network access gateways and strip off the repeated data from the packets. More recently, generic network-level caching architectures have been proposed as alternative to reduce the redundant data traffic in the network, presenting benefits and drawbacks compared to RE. In this paper, we compare a generic in-network caching architecture against state-of-the-art redundancy elimination (RE) solutions on real network topologies, presenting the advantages of each technique. Our results show that in-network caching architectures outperform state-of-the-art RE solutions across a wide range of traffic characteristics and parameters.**

I. Introduction

The fast growth of Internet bandwidth usage, mainly due to the exponential increase in Internet videos (YouTube) and IPTV, has put the Internet infrastructure under high pressure. According to a Cisco survey [9], by 2014 the network traffic is expected to approach 64 Exabytes per month, with videos accounting for more than 91% of global traffic. Redundancy elimination (RE) techniques have been proposed to handle the huge amount of data in the access networks. Their main aim is to remove requests and/or responses of redundant data in the network, reducing the traffic and costs in the access network.

RE techniques can be classified into two kinds: (a) caching to remove transfers, and (b) data replacement with a shim header. Former relies on caching network-level objects and storing them temporarily in the network. Caching techniques rely on redundancy of the traffic [1], [4], implying that a large portion of the network traffic is duplicated and could be cached for later requests. Another incentive is that storage prices have decreased faster than bandwidth costs [11].

The second approach replaces redundant data with a shim header in an upstream middlebox (usually close to the server) and reconstructing it in a downstream middlebox before delivering it to the client. Commercial products provide WAN optimization mechanisms through RE in enterprise networks [6], [13], [19]. Recently, RE has received considerable attention from the research community [3]–[5], [24]. In [3], the authors propose a network-wide approach for redundancy elimination through deployment of routers that are able to remove redundant data in ingress routers and reconstruct it in egress routers. However, they also require tight synchronization between ingress and egress routers in order to correctly reconstruct the packet and they also require a centralized entity to compute the redundancy profiles. In [24], the authors propose to use caches in the local host and use prediction mechanisms to inform servers that they have already the following redundant data. However, they are not able to share the cached data among other nodes due to the local characteristic of the cache.

Although both caching and RE have been around in the research community, there has not been any thorough comparison in the effectiveness of the two above-mentioned strategies: in-network caching vs. redundancy elimination. Work in [16] combines in-network caching and RE, but limiting the applicability of the solution to a single content source only.

In this paper, we perform a comparison between an in-network caching architecture (INCA) and state-of-the-art RE solutions. Although INCA models a generic network caching architecture, it is effectively CCN-like [12], [22]. However, as we want to understand the performance differences between caching and RE, we do consider low-level protocol details.

We perform an extensive comparison, using real network topologies from Rocketfuel, between INCA and RE. We have implemented the different solutions on our testbed and compare them by running them on real network topologies. We consider the position of a single ISP interested in reducing its traffic both within and outside of its own network.

Our key findings can be summarized as follows:

- In terms of reducing external network traffic, INCA is always superior when compared to ISP-internal RE solutions [5]. End-to-end RE solutions [3], [24] can reduce external traffic, but are outside the control of the ISP; furthermore, they are not as effective as INCA.
- In terms of reducing internal network traffic, INCA is in most cases clearly superior to state-of-the-art RE solutions [5], with at least 50–65% improvements in internal traffic reduction.

The organization of this paper is as follows. Section II presents the background information and related work about in-network caching and redundancy elimination solutions. Section III introduces the in-network caching architecture (INCA), describing its main features. Section IV presents the evaluation methodology, and the comparison results between INCA and RE solutions. Finally, Section V summarizes the paper.

II. Background

A. Caching

Recently, information-centric networking (ICN), e.g., [12], [14], [17] has emerged as a more general, network-wide

caching solution. In ICN, content caches in the network (e.g., in routers) store content that passes through them and if they see requests for the same content, they are able to serve it from their cache. INCA is essentially an ICN architecture, but our intention is not to provide yet-another-ICN-architecture. Instead, INCA simply considers the key features of ICN architectures, namely caching and routing towards some point of origin for content, and ignores practical, low-level protocol details. INCA draws inspiration from CCN [12] and our previous work [22], but does not specify low-level behavior.

Other caching proposals also exist. Cache-and-Forward (CNF) [15] is an in-network caching architecture where routers have a large amount of storage. These routers perform content-aware caching, routing and forwarding *packet* requests based on location-independent identifiers, similar to CCN.

B. Redundancy Elimination

Modern RE schemes use a fingerprint-based data stripping model. Nodes generate a set of fingerprints for each packet in transit, where each fingerprint can be generated over a pre-defined block size. Upon detecting a cached fingerprint, the upstream node replaces the data by a fingerprint and the downstream node replaces the fingerprint with the original data, reducing the overall data transmission over the network. As described in [21], both upstream and downstream nodes need to be strongly synchronized in order to work correctly. A similar approach is presented in [2].

Work in [3], [5] proposes to extend the RE technique to the whole network, i.e., to make RE as a basic primitive for Internet.The main idea is to collect redundancy profiles from the network and use a centralized entity to compute paths between destinations within an ISP with higher RE capabilities. Therefore, data going through these networks have higher RE footprint reduction than going to other paths in the network. Despite the improved RE capacity, it still requires strong synchronization between the upstream and downstream routers in order to work properly.

A third RE approach [24] was recently proposed to overcome the synchronization issue in order to be deployed in data-center networks. As cloud elasticity favors the migration and distribution of work among a set of nodes, it is hard to set up the synchronization between two fixed nodes. Therefore, the main idea of [24] is to create a local cache together with a predictive mechanism to acknowledge already cached data to the server. In this scenario, the service sends a predictive acknowledgement to the server informing that the requested data is already present in the client, thus, removing the redundant data. Despite the improvement over the fixed node requirement, the use of local storage prevents the sharing among other nodes, increasing the overall sharing capacity and hit ratio. Therefore, the RE is not network wide, but for redundant data that may be requested again in the local node.

III. INCA: In-Network Caching Architecture

INCA focuses on the following key aspects of ICN architectures: routing requests for content towards a known point, caching of content, and forwarding responses back to the requesting entity. This model is similar to CCN [12].

A. Basic Model

The basic in-network caching mechanism is performed by a *content router* (CR). A CR is a data forwarder similar to a regular router, but has some internal memory that can be used to store data in transit. Each piece of content has a *chunk ID* as its permanent identifier from a cryptographic hash function. Any CR on the path between a server and clients caches the data in its memory. Further requests can be served by the local copy in the CR. For a further discussion on this model and its limitations, we refer the reader to [23].

B. Caching in CRs

As in [23], we use three admission policies for deciding which content a CR caches.

- **ALL** admits all objects into the storage at the CR. In other words, every object that transits through the CR is taken into storage and another object is possibly evicted. This is the typical behavior of web caches.
- **Cachedbit** [23] sets one bit in the CR header to indicate whether a given piece of content has already been cached or not, preventing duplicated content along the same path. If the path between the client and server is n hops, then a CR will cache the content with probability $1/n$ and once the content is cached, downstream CRs will not cache it, with the exception of the last CR on the path which will always cache it (see Section IV-B for an explanation).
- **Neighbor Search (NbSC)** [23] works like *Cachedbit*, but if a CR encounters a miss, it will query neighboring CRs for that piece of content. CRs periodically exchange Bloom filters of their contents with their neighbor CRs. Please see [23] for details about the size of Bloom filters, exchange frequency, and query radius.

We use Least Recently Used policy to decide what to evict when the storage at the CR is full. The results in [23] showed that a Cachedbit-like admission policy is needed to get good caching performance, but that the addition of NbSC gives a considerable boost in reducing network traffic.

IV. Evaluation & Experimental Results

We chose 4 real-world networks from Rocketfuel [20]: Exodus, Sprint, AT&T and NTT, and performed a set of experiments using different cooperative caching strategies. Table I shows an overview of the networks. All the experiments are performed on our department cluster consisting of Dell PowerEdge M610 nodes. Each node is equipped with 2 quad-core CPUs, 32GB memory, and connected to 10-Gbit network. All the nodes run Ubuntu SMP with 2.6.32 kernel.

Network	Routers	Links	# of POPs
Exodus	338	800	23
Sprint	547	1600	43
AT&T	733	2300	108
NTT	1018	2300	121

TABLE I: Topologies used in experiments

Our focus in comparison was to compare ICN-like in-network caching represented by our INCA architecture with state-of-the-art RE solutions. As points of comparison from the RE space, we selected three solutions. We picked SmartRE [5] because it represents a solution internal to a single ISP, much like INCA could be deployed. As examples of end-to-end RE, we selected EndRE [2] and PACK [24].

We implemented SmartRE on top of our software router architecture and used an LP solver to follow the behavior defined in [5]. For EndRE and PACK, we simply compare the performance numbers from the original papers to the numbers of INCA from our experiments.

A. Methodology

We use 10^5 distinct chunks of 1 KB as our data set in the experiments. The popularity follows Zipf distribution and the popularity of the ith most popular chunk is proportional to $1/i^{\alpha}$; we use 0.7, 0.9 and 1.1 as α value.

Content popularity on the Internet is known to follow a Zipf- or a power law distribution [7], [8]. However, in practice one piece of content would consist of several (tens, hundreds, or thousands of) packets. In many cases, there are strong dependencies between packets, i.e., packets belonging to the same piece of content would often be requested together. We have decided to use only single chunks to represent all packets of a piece of content for the reason of reducing the number of parameters to be explored. There is little available information about the distribution of size of content pieces, making it difficult to plug in convincing size distributions. Because of the strong dependencies, one chunk in our experiment would translate to several packets in the real world, and thus the cache sizes would need to be adapted to match that.

The experiment followed the style in [5]. We placed clients and servers at a POP, to represent all the potential servers or clients behind that POP. CRs were installed on every router. We chose top 20 POPs with highest degree as servers, and rest of the POPs as clients. Exodus has only 10 servers due to its small size. A client keeps requesting the chunks from different servers. We experimented two traffic patterns: constant and a gravity model, similar to [5], but differences were negligible.

The metrics we investigated were:

- **Hit rate:** What fraction of requests was served by CRs. Hit rate in our context measures the savings in external traffic from the providers.
- **Content locality:** We analyzed the number of hops needed to get the content to evaluate how the different algorithms are able to get content close to the users. Average hop bears a relationship to the access latency.
- **Footprint reduction:** Network footprint is the product of the amount of data and the network distance from which the data was retrieved. It measures the amount of internal traffic reduction, i.e., a smaller footprint (larger reduction) means less traffic within the ISP's network. This metric was used also in [5] and forms the basis of comparison between INCA and SmartRE.

Our goal was to see what, if any, performance differences there are between INCA and the RE solutions, and determine the causes of the differences. In order to better understand INCA's behavior, we evaluated it very closely in terms of where it places the content and how it uses it.

Note that hit rate does not apply to any of the RE solutions we used. SmartRE reduces only internal traffic, but has no effect on external traffic; in essence it has zero hit rate. End-to-end RE solutions reduce traffic across the whole network, but an intermediate ISP has no control over it, thus there is no "hit rate" since the ISP must transit all traffic that it sees, although the amount of traffic is less than without an RE solution. Likewise, content locality in SmartRE is subsumed by footprint reduction and locality does not apply to end-to-end RE since all content always comes from the origin, although with eliminated redundancy.

We repeated experiments to eliminate variability in the results. Confidence intervals were very tight even after 5 repetitions and for clarity reasons we do not show them.

In the following, we first present INCA's performance in terms of hit rate and content locality; as discussed above these do not apply to RE solutions. Then we show the comparison between INCA and SmartRE for footprint reduction and compare INCA against end-to-end RE solutions.

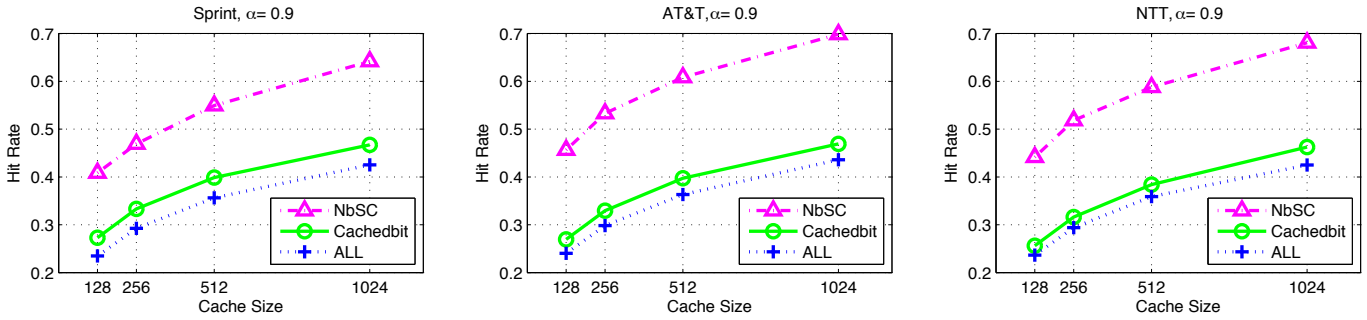
B. Experimental Results

1) Hit Rate: Figure 1 shows the hit rates in three networks we studied. (Exodus yielded similar results and for reasons of space we omit showing them.) On the x-axis we show the number of chunks each CR could store and the y-axis shows the hit rate. Each graph shows 3 curves, one for each admission policy. Recall that we had 10^5 chunks in the experiment, meaning that even with 1000 chunks of storage per CR, one CR can store only about 1% of the total amount of chunks.

Neighbor Search has the highest hit rate and Cachedbit is better than ALL policy. The results here are shown for $\alpha = 0.9$; for $\alpha = 0.7$ or 1.1 the ranking was the same, but the absolute values were lower or higher, respectively.

Even though the networks vary considerably in size, it actually turns out that the network paths between clients and servers are roughly similar in length in all three networks. This means that in all networks the caching capacity on a path is similar, hence getting similar hit rates is to be expected. This is one of the key findings in our work regarding caching performance: *Caching performance of a CR network depends mainly on the path lengths and network topology rather than the absolute number of CRs in the network*. There is some additional evidence in previous studies on Rocketfuel data [10] to suggest that the different networks share some graph theoretical properties. Exactly which properties are important for caching (besides path length) and how they affect performance of caching networks is left for further study.

Both NbSC and Cachedbit show clear gains over the ALL policy. This demonstrates the importance of not wasting storage space as is done by the simple ALL policy, which always admits every chunk into a CR. In contrast, the other two



(a) Sprint (b) AT&T (c) NTT

Fig. 1: Hit rate vs. network topology on POP level.

attempt to ensure that at most one copy of a chunk is created (per path; recall that NbSC uses Cachedbit to decide where to cache). Since both are probabilistic, it is possible, that no copies are created.

2) Content Locality: We also investigated how well the different algorithms were able to get content close to the users. Figure 2 shows the CDF of the number of hops between CRs needed to get the content in the AT&T network with 128 chunks of storage per CR for POP-level and router-level topologies and 512 chunks for POP-level topology. We only plot the line for the Cachedbit strategy. NbSC uses the same placement strategy and ALL was typically slightly worse than Cachedbit. Other networks yielded similar results.

As expected, more storage per CR allows content to be cached closer to the clients. On the router-level topology, the paths are slightly longer, but the overall shape of the curve is similar to POP-level curves.

The slow increase of the CDF indicates that about 30–40% of the cache hits happen in the first 3 POP-level hops. Partial explanation of this is the experiment setup where we have clients behind every non-server POP. This means that a POP that is the second hop for one client is often also a first hop for some other client. In the end, these clients end up fighting for cache space and typically the clients closer to the POP end up getting their most popular content there. Clients for whom that POP is the second or further hop, are therefore less likely to find "new" content there, i.e., content that was not already cached in the first POP. As the paths get longer, this effect seems to wane and the CDF starts a faster increase.

We will discuss footprint reduction below and directly compare INCA with SmartRE.

C. INCA vs. SmartRE

SmartRE [5] uses two network elements, the redundancy profiler and the redundancy-aware route computation element. The redundancy profiler collects in-transit data statistics in order to create a profile of the most popular data to be the ones to be cached in the routers. The redundancy aware route computation computes the paths based on the content stored in the network in order to optimize the redundancy elimination of the network by solving a linear programming (LP) problem. The benefit of such centralized element is the fact that it knows the complete topology and makes it possible to compute a good result for the RE. A totally decentralized SmartRE model is not possible since there must be an entity controlling the synchronization between these points.

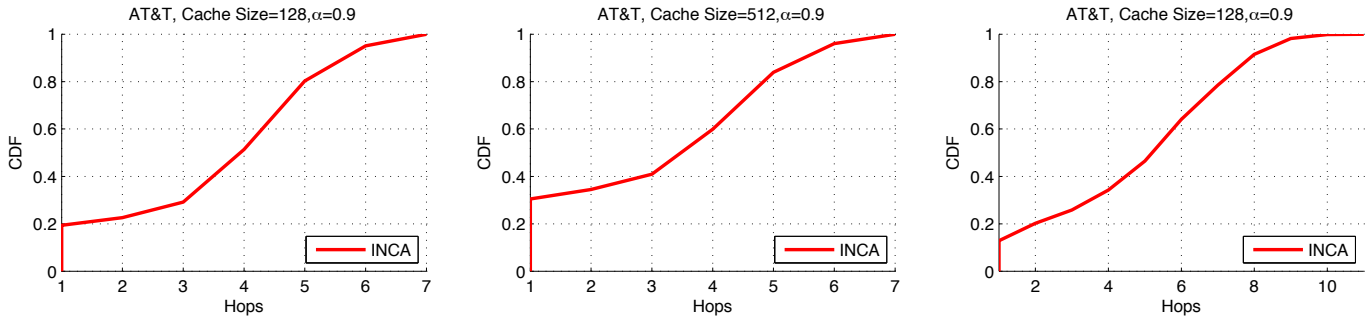
Network	FP Reduction
Exodus	27.55%
Sprint	28.79%
AT&T	31.59%
NTT	30.45%

TABLE II: SmartRE footprint reductions in different networks under ideal conditions

SmartRE reduces the network footprint, because the caches between the ingress and egress store parts of the data and the ingress simply indicates which parts a cache is to substitute in a packet. There is no effect on external traffic. The LP solver knows the redundancy profile of the traffic and calculates a caching manifest which indicates which parts of which packets should be decoded at which caches. There is a very strong link between the total amount of storage in the network and the length of the sampling period which defines how long traffic is observed to compute the redundancy profile. According to [5], sampling periods on the order of a few tens of seconds are to be expected to be reasonable.

We implemented SmartRE on top of our CR testbed. We noticed that SmartRE, or rather the LP defined in [5], is very sensitive to the parameters in the model. Small deviations often lead to large differences in performance, typically for the worse. We were able to determine parameters for what corresponds to the settings in [5] and calculated the footprint reductions for the same traffic as with INCA. These ideal footprint reductions are shown in Table II.

Figure 3 shows the internal traffic reduction as measured by the network footprint reduction. The y-axis shows the fraction of internal traffic that was reduced by the caches in the CRs. As with the other metrics, the differences between the three admission policies are small. Again, NbSC is clearly superior to Cachedbit which, in turn, is clearly superior to the ALL policy. Footprint reduction is the reason why we tweaked Cachedbit to create a copy of the chunk at the CR closest to the client. Without the additional copy, ALL-policy is better at footprint reduction than Cachedbit. We observed that this additional copying drops the hit rate by a negligible amount,



(a) POP-level, 128 chunks (b) POP-level, 512 chunks (c) Router-level, 128 chunks

Fig. 2: CDF of number of hops to content in AT&T network for $\alpha = 0.9$

but raises the footprint reduction considerably.

Contrasting the numbers in Table II with the INCA footprint reductions in Figure 3, we see that they are similar in value. For small INCA cache sizes, SmartRE yields a higher reduction, whereas for larger cache sizes, INCA has the upper hand. However, even for very modest cache sizes, NbSC is able to achieve an equal footprint reduction to SmartRE and for large cache sizes, the footprint reduction is improved by 50–65%. Cooperative caching is therefore much more efficient at reducing internal traffic than SmartRE.

Recall that our INCA experiments considered one chunk to represent one file, whereas in the SmartRE experiments, a chunk is one packet. This means that the footprint reduction numbers cannot be directly compared since traffic is different in the two cases. However, based on the numbers presented in [5], we can infer a mapping between SmartRE and INCA experiments. In [5] it is shown that SmartRE gets close to its ideal performance with 6 GB of storage per router. Assuming the same 6 GB of storage per CR, the case of 1024 chunks of storage, where 1 chunk equals 1 file, would imply the average file size to be about 6 MB. If the content is a mixture of text, images, and short videos, this seems like a reasonable, if not even conservative, number. (For content consisting mainly of larger videos, this would not be sufficient.)

We ran experiments with SmartRE where we took the ideal cache size used to obtain the numbers for Table II, and set it to $1/2$, $1/4$, and $1/8$ of that value. For each case, we then ran the experiment to obtain the reduction in footprint. This allows us to plot the INCA and SmartRE footprint reductions on the same x-axis, shown in Figure 3. This confirms that INCA is more efficient in reducing internal traffic in the network. The additional reduction in traffic varies between almost 200% for small caches and 50% for large caches.

Cachedbit is similar to the heuristic "Heur1" from [5] in how it attempts to place the content. In [5], the performance of these two heuristics was found lacking when compared to the SmartRE algorithm with its centralized controller deciding on what to cache where. If the same translates to an INCA caching network, a centralized controller deciding on placement of chunks in CRs would be a superior choice. However, similar placement problems are often NP-complete [18], although some simplifications are likely to yield a linear program. We have not considered a central placement agent in INCA, although it could be included in future work.

An important difference is that INCA is able to share cache space between clients, whereas SmartRE has fixed buckets for each ingress-egress flow. This gives INCA more possibilities in exploiting the cached data, thus reducing footprint and improving hit rate. *We believe this sharing of cache space between all client and server pairs is what gives INCA an advantage over SmartRE.* Contrasting our results to the single server case presented in [16] is part of our future work.

Comparing INCA with SmartRE, we come to the following conclusions:

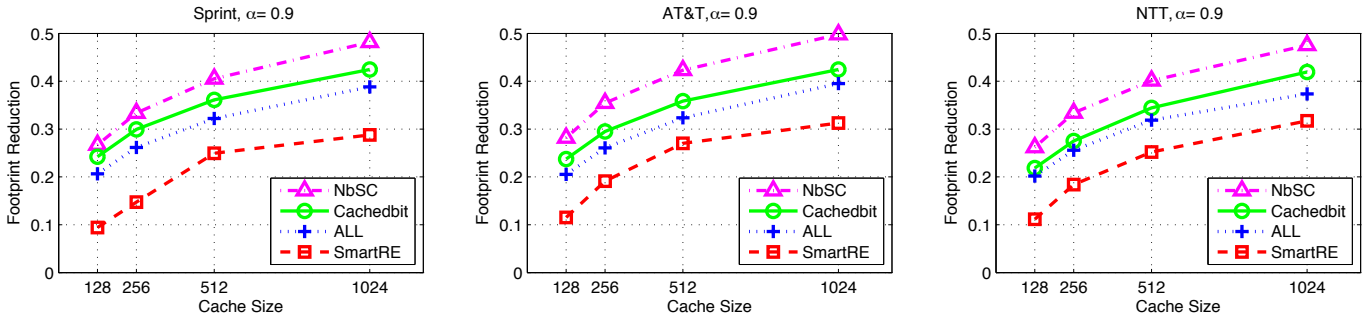
- For external traffic reduction, INCA is always superior, because SmartRE has no effect on external traffic.
- For internal traffic reduction, performance of INCA (with neighbor search) is in most cases clearly superior, up to 50–65% more reduction in internal traffic. However, the differences depend on how the mapping between cache sizes is done and the file size distribution, thus in different environments the results could be different.

However, in our experimental environment INCA with neighbor search is far more effective in reducing both internal and external traffic.

D. INCA vs. End-to-End RE

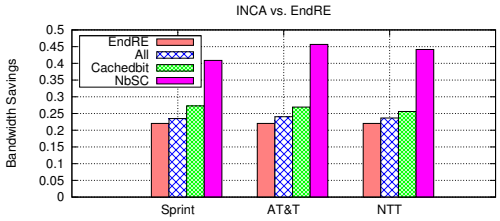
Figure 4 shows the bandwidth savings of both INCA and EndRE [2] on three different networks. We show cache sizes of 128 and 256 chunks. The bandwidth savings of EndRE remains the same on three networks because it is end-to-end solution. The network topology does not affect its performance. We can clearly see that INCA is superior to EndRE. Even the ALL strategy is slightly better than EndRE in all three networks. PACK [24] is another end-to-end RE solution, but according to [24], its performance is about 2% worse than EndRE. Larger cache sizes improve INCA's performance; figures not shown due to space limitations. Note that INCA's savings are a combination of results shown in Figures 1 and 3.

Anand et al. [4] have evaluated real trace captures and their results suggest that a middlebox-based solution (i.e., something akin to INCA) has an advantage over end-to-end RE solutions in saving network bandwidth. INCA does have a definite advantage in not requiring synchronization between

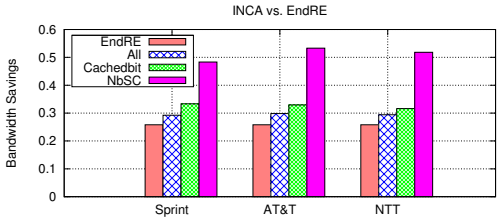


(a) Sprint (b) AT&T (c) NTT

Fig. 3: Comparing footprints of INCA and SmartRE



(a) 128 chunks of storage



(b) 256 chunks of storage

Fig. 4: INCA vs. EndRE

the server and client and since some content can be served from CRs along the path, we avoid having to do a round-trip to the origin of the content, possibly speeding up the transfer.

V. Conclusion

In this paper we have compared in-network caching with standard redundancy elimination solutions in terms of their effectiveness at reducing network traffic load. As an example of in-network caching, we have presented INCA, a caching architecture which aims at capturing the salient features of information-centric networks. We have kept the design of INCA minimal and only consider simple solutions for the problems of caching and routing. Our comparison on Rocketfuel topologies shows that INCA is superior to SmartRE in the ability to reduce external and internal network traffic, with additional reductions of up to 65% in internal traffic. Similar results hold for comparisons against end-to-end RE solutions.

References

[1] Ager, B.; Schneider, F.; Juhoon Kim; Feldmann, A. Revisiting cacheability in times of user generated content. *IEEE INFOCOM*, March 2010.

[2] B. Aggarwal, et al. Endre: an end-system redundancy elimination service for enterprises. In *Proceedings of NSDI*, 2010.

[3] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker. Packet caches on routers: the implications of universal redundant traffic elimination. In *ACM SIGCOMM*, 2008.

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

[5] A. Anand, V. Sekar, and A. Akella. SmartRE: an architecture for coordinated network-wide redundancy elimination. In *ACM SIGCOMM*, 2009.

[6] Bluecoat Mach5. http://www.bluecoat.com/products/mach5.

[7] L. Breslau, P. Cue, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *IEEE INFOCOM*, 1999.

[8] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Trans. Netw.*, 17(5):1357–1370, 2009.

[9] Cisco Virtual Network. http://www.cisco.com/web/cy/solutions/sp/sp_strategy, 2009.

[10] S. Eum, S. Arakawa, and M. Murata. Toward bio-inspired network robustness – step 1. modularity. In *Proc. of Bionetics*, 2007.

[11] IETF Decoupled Application Data Enroute (DECADE) Workgroup. http://datatracker.ietf.org/wg/decade/, 2011.

[12] 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.

[13] Juniper WXC590 Application Acceleration Platform. http://www.juniper.net/us/en/products-services/application-acceleration/wxc-series.

[14] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. *ACM SIGCOMM*, 2007.

[15] Paul, S. Yates, R. Raychaudhuri, D. Kurose, J. The cache-and-forward network architecture for efficient mobile content delivery services in the future internet. *Innovations in NGN: Future Network and Services*, pages 367–374, May 2008.

[16] D. Perino, M. Varvello, and K. P. N. Puttaswamy. Icn-re: redundancy elimination for information-centric networking. In *Proceedings of SIGCOMM ICN workshop*, 2012.

[17] Publish/Subscribe Internet Routing Paradigm. Conceptual architecture of psirp including subcomponent descriptions. Deliverable d2.2, PSIRP project. , August 2008.

[18] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *IEEE Infocom*, 2001.

[19] Riverbed Wan Optimization. http://www.riverbed.com/us/solutions/wan_optimization.

[20] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with rocketfuel. In *Proceedings of ACM SIGCOMM*, 2002.

[21] N. T. Spring and D. Wetherall. A protocol-independent technique for eliminating redundant network traffic. In *ACM SIGCOMM*, 2000.

[22] W. Wong, M. Magalhães and J. Kangasharju. Content Routers: Fetching Data on Network Path. *IEEE ICC*, 2011.

[23] W. Wong, L. Wang, and J. Kangasharju. Neighborhood Search and Admission Control in Cooperative Caching Networks. *IEEE GLOBECOM*, 2012.

[24] E. Zohar, I. Cidon, and O. O. Mokryn. The power of prediction: cloud bandwidth and cost reduction. In *ACM SIGCOMM*, 2011.