Stream Reasoning-Based Control of Caching Strategies in CCN Routers

Harald Beck*, Bruno Bierbaumer†, Minh Dao-Tran*, Thomas Eiter*,
Hermann Hellwagner† and Konstantin Schekotihin†
* TU Wien, Vienna, Austria, Email: {beck,dao,eiter}@kr.tuwien.ac.at
†Alpen-Adria-Universität Klagenfurt, Austria, Email: bruno@itec.aau.at, firstname.lastname@aau.at

***Abstract*—Routers in Content-Centric Networking (CCN) may locally cache frequently requested content in order to speed up delivery to end users. Thus, the issue of caching strategies arises, i.e., which content shall be stored and when it should be replaced. In this work, we employ, and study the feasibility of, novel techniques towards intelligent control of CCN routers that autonomously switch between existing caching strategies in response to changing content request patterns. In particular, we present a router architecture for CCN networks that is controlled by rule-based stream reasoning, following the recent formal framework LARS which extends Answer Set Programming for streams. The obtained possibility for flexible router configuration at runtime allows for versatile network control schemes and may help advance the further development of CCN. Moreover, the empirical evaluation of our feasibility study shows that the resulting caching agent may give significant performance gains.**

I. Introduction

The Internet evolved from a small research network that focused on sending text messages, to a global network of content distribution; Cisco estimates that by 2019, 80% of the Internet traffic will be video content [1]. Commercial Content Distribution Networks (CDNs), built as overlays on the traditional Internet architecture, have been developed to cope with today's content delivery demands. In general, though, today's Internet architecture does not fit content-heavy applications well that have evolved over the decades [2].

In response to this, various Future Internet research efforts are being pursued, among them Information-Centric Networking (ICN) [3], and in particular *Content-Centric Networking* (CCN) [4] whose concepts are being taken further by the *Named Data Networking* (NDN) project [5]. As we are concerned with the basic CCN router functionality only, we will subsequently just refer to the CCN approach.

CCN attempts to replace the current location-based addressing with a name/content-based approach. That is, data packets shall be routed and retrieved based on *what* the user wants, not from *where* it is retrieved. In a sense, CDNs do provide this, yet CCN supports this at the network level by making content identifiable.

An important element of the CCN architecture is that every CCN router has a *cache* (content store) which holds content items that were recently transmitted via the router. A request for a content item may be satisfied by a router rather than routed to the original content source; thus data is delivered to end users faster. A *caching strategy* defines which content is stored, on which routers, and for how long before being replaced. There is a rich literature of strategies and mechanisms for ICN/CCN [6], [7], [8], [9], with a variety of parameters influencing the overall behavior of a network.

The caching strategies can be roughly classified into adaptive and reactive ones. The adaptive strategies use information about interests of users saved by a network logging system. This information is then used to estimate popularity of content in the future and push it to the caches of routers. Therefore, adaptive strategies are mostly used in CDNs which, by their nature, are tightly integrated with the networks of content providers. Strategies used in CCNs are essentially reactive, i.e., they use a kind of heuristic to predict whether a forwarded content chunk might be interesting for other users. If so, the chunk is added to the router's cache. Some of the reactive strategies go even further and allow for synchronization of caching decisions between multiple routers. For instance, the most popular content must be cached by routers of the lowest levels in the network topology. Such strategies, however, often work only for specific topologies, like trees.

Recent evaluations of CCN caching strategies, like [6], indicate that no strategy is superior in all tested scenarios. Furthermore, selecting a good caching strategy and fine-tuning its parameters is difficult, as the distribution of the interests of consumers in content may vary greatly over time [10], [11].

Example 1 Consider a scenario in which some music clips get very popular over a short period of time. In this case, network managers may manually configure the routers to cache the highly popular content for some time period, and then to switch back to the usual caching strategy when the interests get more evenly distributed. Since this time period is hard to predict, it would be desirable that routers autonomously switch their caching strategies to ensure high quality of service. ■

As real CCNs are not deployed yet, there is currently no real-world experience to rely on, and developing control methods for caching strategies is not well supported.

Motivated by this, we suggest and investigate a router architecture allowing for dynamic switching of caching strategies in reaction to the current network traffic, based on *stream rea-*

This work was partly funded by the Austrian Science Fund (FWF) under Project "Distributed Heterogeneous Stream Reasoning" (P26471), Doctoral Programme (DK) "Logical Methods in Computer Science" (W1255-N23), and CHIST-ERA Project "A Context-Adaptive Content Ecosystem Under Uncertainty" (I1402).

soning, i.e., reasoning over recent snapshots of data streams.

Contributions. Our contributions are summarized as follows.

(1) We present an *Intelligent Caching Agent* (ICA) for the control of caching strategies in CCN routers using stream reasoning, with the following features:

- ICA extends a typical CCN architecture with a decision unit, resulting in the first implementation of a local and dynamic selection of an appropriate caching strategy.
- The main component of the decision unit is based on the rule-based stream reasoning framework LARS [12], which is an extension of Answer Set Programming (ASP) for streams (see Section III for details). LARS enables network managers to control caching strategy selection in a concise and purely *declarative* way. Furthermore, the selection control can be modified without taking a router offline, which is another important criterion for such systems.

(2) To support the research and testing of dynamic cache strategy selection, we suggest an extension of ndnSIM [13] – a popular CCN simulator – for iterative empirical assessment of proposed solutions for intelligent administration. In particular, the resulting toolset is designed to: (i) simulate various CCN application scenarios, (ii) implement different architectures of CCN routers, (iii) apply rule-based stream reasoning to make decisions about the caching strategy configuration for every router in the network, (iv) react quickly to inferred information from continuously streaming data, and (v) be expressible in an understandable and flexible way for fast experimentation.

(3) We provide a detailed evaluation of our methods on two sample scenarios in which content consumers unexpectedly change their interests, as in Example 1. Our results indicate a clear performance gain when basic caching strategies are dynamically switched by routers in reaction to the observed stream of requested data packets.

In summary, we provide a feasibility study for using logic-based stream reasoning techniques to guide selection of caching strategies in CCNs. Moreover, we also provide a detailed showcase of analytical, declarative stream reasoning tools for Artificial Intelligence-supported network control. To the best of our knowledge, no similar work exists to date. We envision that this approach could prove beneficial beyond CCN, e.g., for today's SDN controllers.

It must be emphasized that our approach (the ICA) operates as a *high-level* controller/manager (of *given* caching strategies in routers). Such high-level control schemes can be expressed *declaratively* as rules and reasoned about, as shown in the course of the feasibility study throughout the paper (Examples 2–4 and Program P in Fig. 1). Low-level control such as fine-tuning or dynamically learning parameters of the individual caching strategies, or even developing and analyzing new ones, is clearly beyond the scope of the approach. This situation is akin to the robot control discussed in [14]: high-level control, exerted by a knowledge-based system, represents the robotic domain and environment, and reasons about and plans the general actions of the robot, while low-level control, realized by probabilistic graphical models, senses the environment and implements the robot's actuation in the real world.

The specific parameters of the feasibility study were chosen as to evoke interesting behavior in our synthetic test scenarios, while being realistic according to the literature on caching in CCN (cf. Sect. IV).

II. Preliminaries

Content-Centric Networking. The operation of a CCN network relies on two packet types, *Interest* and *Data* packets. Clients issue *Interest* packets containing the *content name* they want to retrieve. CCN routers forward the *Interest* packets until they reach a content provider that can satisfy them with the content addressed by the *content name*. The content provider answers with a *Data* packet which travels back to the original content consumer following the previous *Interest* packets. In addition to delivering the *Data* packets back to the consumer, the CCN routers have the possibility to cache these packets in their *Content Stores*. Thus, the *Interest* packets of another consumer can be directly satisfied out of a *Content Store* without the need of going all the way to the original content provider. These caches make it possible to keep popular content near the consumer, satisfy content requests directly out of caches and reduce the network load [4].

Content Popularity Distribution. Not all content is equally popular. Usually, there is a small number of very popular content items and lots of unpopular ones, which is described in the literature with a *Zipf* distribution [15]. Let C be a number of items in the content catalog, α be a value of the exponent characterizing the distribution and i be a rank of an item in the catalog. Then, a Zipf distribution predicts the frequency of Interest packets for item i as [16]:

$$P(X = i) = \frac{1/i^{\alpha}}{\sum_{j=1}^{C} 1/j^{\alpha}} \tag{1}$$

The variation of the exponent α allows to characterize different popularity models for contents requested by consumers: (i) if α is high, the popular content is limited to a small number of items; (ii) if α is low, every content is almost equally popular.

The content popularity distribution and its exponent α can be estimated by counting the *Interest* packets arriving at a router. The estimated values $\hat{\alpha}$ of the α parameter can be used to form rules like: "If a small number of content items has been very popular ($\hat{\alpha} \geq 1.8$) for the last 5 minutes, then action C should be applied."

Caching Strategies. In our feasibility study, a caching strategy decides which item gets replaced in the full cache storage if a new item should be added; in other words, we are concerned with cache *replacement* strategies. We consider the following strategies [13]:

- *Least Recently Used.* The LRU strategy keeps the cached items in a list sorted by their access time stamps and replaces the oldest item.
- *First-In-First-Out.* For the FIFO strategy, the cache is implemented as a simple queue and replaces the earliest inserted item.

• *Least Frequently Used.* The LFU strategy counts how often an item in the cache is accessed. When caching a new item, the item with the smallest access count is replaced.
• *Random.* The Random strategy replaces a random item in the cache with a new one.

III. Intelligent Caching Agent

In this section we present the Intelligent Caching Agent (ICA), which is an extension of a typical CCN router with a rule-based stream reasoning decision unit. Stream reasoning [17] emerged from stream processing for real-time reasoning about data streams. Initially, the focus was on *continuous* queries similar to SQL [18], [19]. Later works also dealt with advanced logic-oriented reasoning [20], [21], [22], [23], [12] on streaming data.

To the best of our knowledge, stream reasoning techniques have not yet been considered in CCN. Research on CCN so far addressed various challenges involving hardware, architecture, data formats, protocols, and security, among others. We argue that, from an information-oriented point of view, CCN is to a large degree a task of stream processing. In particular, the intelligent cache control of routers that we concentrate on in this work, adds the need to logically reason over the streaming data in real-time.

Example 2 Consider the following rules to select a caching strategy. If in the last 30 sec. there was always a high $\hat{\alpha}$ value (some content is very popular), use LFU, and for a medium value, take LRU. Furthermore, use FIFO if the value is low and at least once in the last 20 sec. 50% of traffic was real-time content. Otherwise, use Random. ∎

Example 2 illustrates that caching strategy control can greatly be simplified by a fully declarative, rule-based language. Similar languages are widely used, for instance, in administration of firewalls, routing, audit, etc. [24].

Notably, envisaged real-world deployments of CCNs will involve much more complex rules, where advanced reasoning features will be beneficial. This includes declarative exception handling, reasoning with multiple models, defaults, and the possibility to adjust the involved logic in a flexible and modular way that allows for easy refinements.

LARS. LARS [12] is a recently suggested language based on Answer Set Programming (ASP) [25], which provides operators to deal with stream-specific information; i.e., access to temporal information and the possibility to limit reasoning to recent *windows* of data. Such recent snapshots of data can also be processed using query languages like CQL [19], but complex queries quickly become unreadable and are less modular than rule-based approaches. For instance, selection of a strategy in Example 2 would require either to write nested queries or to include another layer in the architecture where separated query results are compared. In contrast, the LARS language allows to use simple exchangeable rules to implement strategy selection criteria. Therefore, stating decision-making processes as small "if-then" statements is more natural than encoding them in (often complex) SQL queries. Furthermore, LARS supports updates at runtime thus allowing for manual interventions in novel network situations. The essence of new situations, as understood by human network managers, can be added to the existing set of rules without the need for restarting the system. Furthermore, LARS as such provides a high degree of expressivity and is suitable for more involved setups which may build on the feasibility study presented in this paper.

Syntax. A LARS program is a set of rules of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_j, \mathrm{not}\ \beta_{j+1}, \ldots, \mathrm{not}\ \beta_n \qquad (n \geq 0)$$

where $\alpha, \beta_1, \ldots, \beta_n$ are formulas as described below and not denotes *negation-as-failure*. That is, $\mathrm{not}\ \beta_i$ is true if it cannot be shown that β_i is true, i.e., there is no justification for β_i.

LARS formulas are defined as follows: Let a be an atom, i.e., an elementary propositional statement that can be either true or false, and $t \in \mathbb{N}$. The set $\mathcal{F}$ of *formulas* is defined by the grammar $\varphi ::= a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \boxplus^w\varphi \mid \Diamond\varphi \mid \Box\varphi \mid @_t\varphi$. Besides negation, conjunction, disjunction and implication, LARS formulas may comprise the following operators:

- *window operator* $\boxplus^w$ limits evaluation of a formula φ to the temporal range $[t-w, t]$, where t is the current time;
- *temporal quantification*, denoted $\Diamond$ and $\Box$, is used to query whether a formula φ holds at *some* time point in a selected window, or at *all* time points, respectively;
- *temporal specification* operator $@_{t'}$ allows to 'jump' to a specific time point t' and to evaluate a formula φ at t'.

Example 3 The implication $\varphi = \boxplus^{30}\Box high \rightarrow use(lfu)$ informally specifies the following: if in the last 30 sec. ($\boxplus^{30}$) the predicate $high$ always ($\Box$) holds, use lfu. ∎

Example 4 Fig. 1 formalizes the rules given in Example 2. In this LARS program, the atom $\hat{\alpha}(V)$ is used to retrieve from the router an estimation of the α value V. Similarly, $rtm50$ is true if at least 50% of the content forwarded by the router was real-time. Rules $(r_1), (r_2)$ and (r_3) are used to derive $high, mid$ and low for each time T, measured in sec., in the selected interval of the last 30 sec., if the estimated value of the parameter α is above 1.8, between 1.2 and 1.8 and below 1.2, respectively. Note, we use integers to represent real values as required by our LARS implementation and expressions like "$V \geq 18$" is familiar infix notation for predefined atoms of the form $\geq (V, 18)$.

Next, rules $(r_4), (r_5)$ and (r_6) are used to derive commands to the router and are already discussed in Example 3. Finally, the default application of the random strategy is decided by the rules (r_7) and (r_8). Thus, (r_7) derives $done$ if one of the strategies is selected. $use(random)$ is derived in case when $done$ cannot be derived, i.e., none of the strategies is selected.∎

System Description. As shown in Fig. 2, an Intelligent Caching Agent (ICA) extends the architecture of a common CCN router comprising a *networking unit* with a number of communication interfaces. This unit is responsible for the basic

$$r_1: \ @_T high \leftarrow \boxplus^{30} @_T \hat{\alpha}(V),\ V \geq 18. \qquad r_5: \ use(lru) \leftarrow \boxplus^{30} \Box mid.$$
$$r_2: \ @_T mid \leftarrow \boxplus^{30} @_T \hat{\alpha}(V),\ 12 \leq V < 18. \qquad r_6: \ use(\textit{fifo}) \leftarrow \boxplus^{30} \Box low,\ \boxplus^{20} \Diamond rtm50.$$
$$r_3: \ @_T low \leftarrow \boxplus^{30} @_T \hat{\alpha}(V),\ V < 12. \qquad r_7: \ done \leftarrow use(\textit{lfu}) \vee use(lru) \vee use(\textit{fifo}).$$
$$r_4: \ use(\textit{lfu}) \leftarrow \boxplus^{30} \Box high. \qquad r_8: \ use(random) \leftarrow \text{not } done.$$

Fig. 1: Program P deciding which caching strategy to use

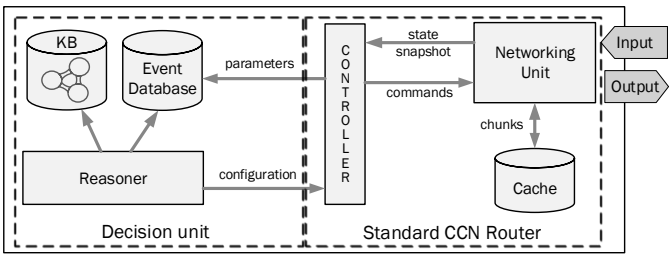


Fig. 2: Architecture of an Intelligent Caching Agent (ICA)

functionality of the router such as processing, forwarding of packets, etc. The networking unit is monitored by a *controller*, which implements various supervising functions including a number of caching strategies. The *decision unit* of an ICA consists of three main components: (1) an event database (DB) storing snapshots of parameters observed by the controller, (2) a knowledge base (KB) containing the ICA logic and (3) a reasoner that decides about configuration of the controller given the KB and a series of events in the DB. The components (2) and (3) are based on the LARS framework described above.

Simulation Environment. We implemented our ICA approach by extending the CCN simulator ndnSIM 2.0 [13]. Thus, we added a Content Store Tracer component to observe states of the router components of the simulator and push this data to the event database. Similarly to [26], our extension of ndnSIM periodically triggers the LARS-based solving process for a decision about the controller configuration based on the events stored in the database as well as functions estimating values of α and $rtm50$.

IV. Evaluation

We now present the evaluation of the resulting simulation system presented in Fig. 2. We show the applicability of our architecture for dynamic caching strategy control and demonstrate the potential performance gains over static caching approaches. For a more detailed version see [27].

A. Setup

We selected the *Abilene* network from the Rocketfuel project [28], which has plausible properties of a future CCN network [29]. For every simulation run (see below), we connected each of the 1000 consumers uniformly at random to one of the 11 routers. All content providers were connected to exactly one router.

Scenarios. We used two *scenarios* to test reaction to content popularity changes:

- *LHL* starts with a low α value, then changes to a high value, and then back to low.
- *HLH* conversely starts with a high α value, changes then to low and back to high.

Although the parameter α content popularity is central to the caching performance, there is no consent in the CCN literature about the exact value. We take the extremal values found in [29], [16], [7], i.e., $\alpha = 0.4$ (Low) and $\alpha = 2.5$ (High) .

The total time span of each simulation is 1800 secs, and we switch the value of α after 600 and 1200 secs. In each of these 600-secs intervals, each consumer starts downloading a video at a time point selected uniformly at random.

Caching Strategies. Recall that our goal is here neither a study of given caching mechanisms as such, nor the development of a new static caching strategy. Instead, we are interested in evaluating (i) our architecture for flexible configuration based on stream reasoning techniques, and (ii) our hypothesis that switching between strategies locally depending on the situation may lead to better performance.

To this end, we employ the (static) strategies *Random* and *LFU* for experimentation [7] in two dynamic settings, where different strategies switch between them flexibly. The hypothetical *Admin* strategy is manually configured, in line with the experimentation setup. Here, all routers change their caching strategy exactly at a phase change from L to H or vice versa. When α is low (L phase), Random is used on all routers, else LFU. The *Intelligent Caching Agent (ICA)* strategy does not enforce the same strategy for all routers and switches the caching mechanism based on locally observed data streams.

Simulation System Parameters. We have analyzed the typical setups in the literature [29], [4], [30], [31] and selected the following parameters: in each session of 1800 sec., 1000 users might query 50 videos, where every video is split into 1000 chunks of 10 kB each. Since the literature disagrees on the sizes of router caches, we executed every experiment with different sizes including caches for 50, 250, 500, 2000 and 5000 chunks. That is, every router could store 0.1, 0.5, 1, 4 or 10% of all available chunks.

Performance Metrics. We use two metrics that are typically used in the CCN community for evaluating caching mechanisms [6]. First, the *cache hit ratio* is the number of cache hits per total number of requests. A *cache hit* is counted when an *Interest* packet can be satisfied by some router's *Content Store*. Second, the *cache hit distance* is the average number of *hops* for a *Data* packet, i.e., the number of routers travelled between the router answering a request and the consumer that

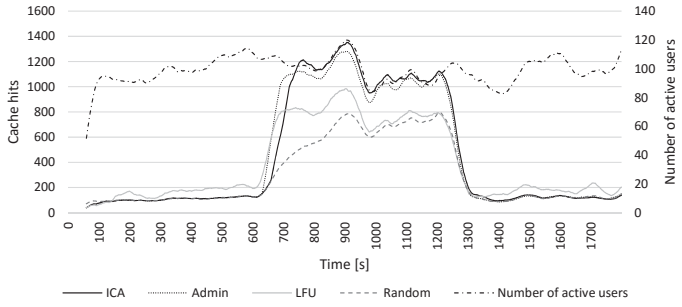


Fig. 3: Cache hits in simulation run for LHL

had issued it. We aim for a high cache hit ratio and a low cache hit distance.

B. Results

Due to space constraints, we omit the first part of the evaluation in which we determined 1% to be a reasonable cache percentage that is large enough to measure benefits by intelligent caching, yet small enough to be realistic, given the hardware constraints of potential future real-world deployments (cf. [27]). We focus here on the performance analysis, for which we ran 30 tests for every combination of the 4 caching strategies (Random, LFU, Admin, ICA) and 2 scenarios (LHL and HLH), i.e., a total of 240 runs.

Reacting to Changing Content Popularity. Fig. 3 shows the development of the cache hits over time during a single simulation run for Scenario LHL. The number of active users (right y-axis) initially increases rapidly and it then varies slightly at about 95.

The cache hit increase after 600 sec. is observed for all caching strategies. Conversely, one can see the reverse effect after switching back to a more equal content interest distribution after 1200 sec. The Random strategy is in general slowly responding to the new situation and shows a steady increase in the middle H phase compared to the rapid increase of hits seen with the other strategies. Note that the Random strategy stores and replaces arbitrary chunks. If requested content becomes less equally distributed, the stored chunks tend to be more often those that are more popular. This explains why cache hits are increasing also under Random and why its reaction is slower.

LFU reacts well to the change from L to H but still has to deal with the recent history of cache items gathered in the phase with low α. Thus, it does not achieve as high hit rates as the alternating cache strategies Admin and ICA.

It is no surprise that the hypothetical, manual Admin strategy shows very fast adaptation in both situations where the value α changes. Notably, the reactive ICA strategy shows about the same performance as Admin.

Interestingly, up to 5 routers used the Random strategy in the H phase under the ICA strategy. At this point, the benefit of dynamic and *local* caching strategy control becomes evident: while the global content popularity is configured to be high ($\alpha = 2.5$), the local estimates $\hat{\alpha}$ of some routers might be different. Here, intelligent control by ICA can account for differences arising from topological effects.

Performance Comparison. Fig. 4 presents an overview of our performance comparisons, indicating the performance of caching strategies LFU, Admin and ICA in relation to Random, which is used as baseline (100%). All box plots visualize the aggregated results over 30 individual runs with the respective caching strategy.

Cache Hit Ratios. Figure 4a compares the cache hit ratios of all strategies in scenario LHL. We see that LFU has a higher variance than Random and is slightly worse on average. Both dynamic strategies outperform the static ones.

Figure 4b depicts the converse scenario HLH. In the two H phases, which comprise two thirds of the overall runtime, LFU works well and thus the ratios are closer to each other than in LHL. Still, ICA is performing better than the other strategies, even compared to Admin. Here, the benefit of separate strategies for routers becomes visible.

Cache Hit Distances. Similarly as for cache hit ratios, Figures 4c and 4d show the aggregated cache hit distances for for scenarios LHL and HLH, respectively.

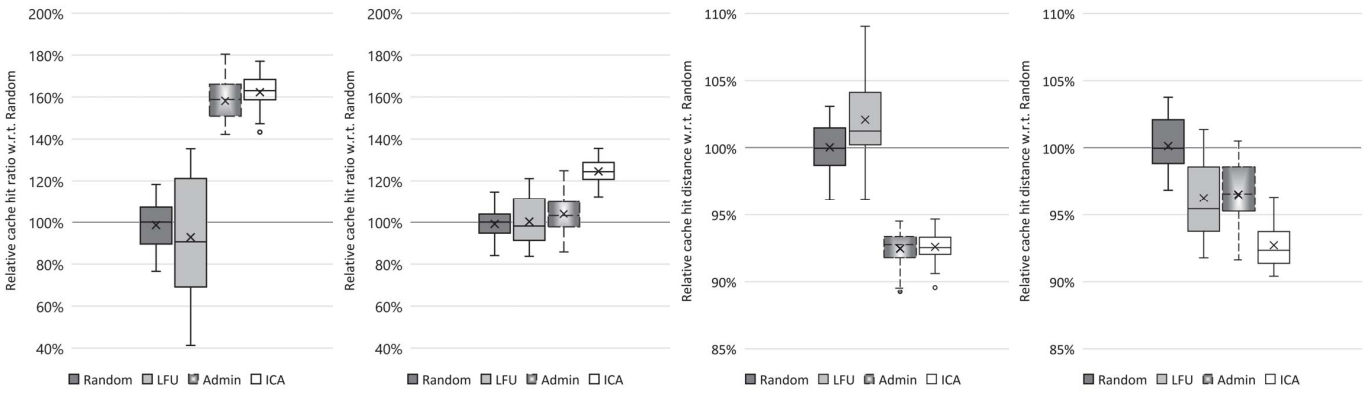
Also according to this metric, Admin and ICA deliver better performance than the static approaches. Figure 4c again shows a clear difference between static and alternating strategies, in terms of mean values and of variance. Figure 4d confirms the better performance of LFU compared to Random for the Scenario HLH. As for cache hit ratio, LFU is also close to the Admin strategy in terms of cache hit distance. Finally, as above, ICA is even better than Admin due to its flexibility.

In summary, dynamic switching is advantageous in both settings. ICA is at least as good as Admin in LHL, and proves to be the best strategy for HLH. Note, both dynamic strategies lead to a decreased cache hit distance relative to the Random strategy, eventually resulting in lower content access delays.

V. Conclusion

We presented a feasibility study of how rule-based reasoning techniques can be used for adaptive network control tasks that depend on streaming data. More specifically, we provided an architecture for the dynamic selection of caching strategies in future CCN routers. For the proof of concept in this study, we focused on selecting the cache replacement strategies in the CCN routers. Our empirical evaluations indicate that dynamic switching of caching (cache replacement) strategies in reaction to changing user (content access) behavior may yield performance gains.

We focused on a principled approach of automated decision making by means of high-level reasoning on stream data and provided a purely declarative control unit. We developed and presented a toolset that combines the CCN simulator ndnSIM 2.0 with a control and decision unit based on the formal framework LARS that extends Answer Set Programming for stream reasoning. Depending on parameters and events monitored from the CCN network simulator, our Intelligent Caching



(a) LHL cache hit ratio (b) HLH cache hit ratio (c) LHL cache hit distance (d) HLH cache hit distance

Fig. 4: Aggregated evaluation results over 30 runs for each caching strategy

Agent periodically triggers the LARS-based reasoning process to dynamically control the local router's caching strategy.

Full-scale industrial solutions will involve much more complex decision rules and processes, and it remains to be studied how well the architecture and in particular stream reasoning tools will work in complex routers and traffic situations, and how the approach can be exploited for protocol design.

REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2014-2019," *White Paper*, 2016.

[2] M. Handley, "Why the Internet Only Just Works," *BT Technology Journal*, vol. 24, no. 3, pp. 119–129, Jul. 2006.

[3] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. Braynard, "Networking Named Content," in *Proc. CoNEXT*, 2009, pp. 1–12.

[5] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM SIGCOMM Comp. Comm. Rev.*, vol. 44, no. 3, pp. 66–73, 2014.

[6] M. Zhang, H. Luo, and H. Zhang, "A Survey of Caching Mechanisms in Information-Centric Networking," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 3, pp. 1473–1499, 2015.

[7] S. Tarnoi, K. Suksomboon, W. Kumwilaisak, and Y. Ji, "Performance of Probabilistic Caching and Cache Replacement Policies for Content-Centric Networks," in *Proc. IEEE LCN*, 2014, pp. 99–106.

[8] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity-based and Collaborative In-network Caching for Content-oriented Networks," in *IEEE INFOCOM Workshops*, 2012, pp. 316–321.

[9] C. Bernardini, T. Silverston, and O. Festor, "MPC: Popularity-based Caching Strategy for Content Centric Networks," in *Proc. IEEE ICC*, 2013, pp. 3619–3623.

[10] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding User Behavior in Large-scale Video-on-Demand Systems," in *Proc. EuroSys*, 2006, pp. 333–344.

[11] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. B. Moon, "Analyzing the Video Popularity Characteristics of Large-scale User Generated Content Systems," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1357–1370, 2009.

[12] H. Beck, M. Dao-Tran, T. Eiter, and M. Fink, "LARS: A Logic-based Framework for Analyzing Reasoning over Streams," in *Proc. AAAI-15*, 2015, pp. 1431–1438.

[13] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A New Version of the NDN Simulator for NS-3," NDN, Technical Report NDN-0028, 2015.

[14] S. Zhang, M. Sridharan, M. Gelfond, and J. Wyatt, "KR3: An Architecture for Knowledge Representation and Reasoning in Robotics," in *Proc. NMR*, 2014.

[15] D. Rossi and G. Rossini, "On Sizing CCN Content Stores by Exploiting Topological Information," in *Proc. IEEE INFOCOM Workshops*, 2012, pp. 280–285.

[16] G. Rossini and D. Rossi, "A Dive into the Caching Performance of Content Centric Networking," in *Proc. IEEE CAMAD*, 2012, pp. 105–109.

[17] E. Della Valle, S. Ceri, F. van Harmelen, and D. Fensel, "It's a Streaming World! Reasoning upon Rapidly Changing Information," *IEEE Intelligent Systems*, vol. 24, no. 6, pp. 83–89, 2009.

[18] S. Babu and J. Widom, "Continuous Queries over Data Streams," *ACM SIGMOD Record*, vol. 3, no. 30, pp. 109–120, 2001.

[19] A. Arasu, S. Babu, and J. Widom, "The CQL Continuous Query Language: Semantic Foundations and Query Execution," *VLDB J.*, vol. 15, no. 2, pp. 121–142, 2006.

[20] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele, "Engineering an Incremental ASP Solver," in *Proc. ICLP*, 2008, pp. 190–205.

[21] M. Gebser, T. Grote, R. Kaminski, P. Obermeier, O. Sabuncu, and T. Schaub, "Stream Reasoning with Answer Set Programming. Preliminary Report." in *Proc. KR*, 2012, pp. 613–617.

[22] C. Zaniolo, "Logical Foundations of Continuous Query Languages for Data Streams," in *Proc. Datalog 2.0*, 2012, pp. 177–189.

[23] A. Mileo, A. Abdelrahman, S. Policarpio, and M. Hauswirth, "StreamRule: A Nonmonotonic Stream Reasoning System for the Semantic Web," in *Proc. RR*, 2013, pp. 247–252.

[24] E. Nemeth, G. Snyder, T. R. Hein, and B. Whaley, *UNIX and Linux System Administration Handbook*, 4th ed. Prentice Hall, 2010.

[25] G. Brewka, T. Eiter, and M. Truszczyński, "Answer Set Programming at a Glance," *Comm. of the ACM*, vol. 54, no. 12, pp. 92–103, 2011.

[26] T. M. Do, S. W. Loke, and F. Liu, "Answer Set Programming for Stream Reasoning," in *Proc. Canadian AI*, 2011, pp. 104–109.

[27] H. Beck, B. Bierbaumer, M. Dao-Tran, T. Eiter, H. Hellwagner, and K. Schekotihin, "Stream Reasoning-Based Control of Caching Strategies in CCN Routers," Tech. Rep., 2016. [Online]. Available: https://drive.google.com/file/d/0B1USOxZtZtSFME96MVRDT2FXVUE

[28] N. T. Spring, R. Mahajan, D. Wetherall, and T. E. Anderson, "Measuring ISP Topologies with Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, 2004.

[29] D. Rossi and G. Rossini, "Caching Performance of Content Centric Networks under Multi-path Routing (and More)," *Relatório técnico, Telecom ParisTech*, 2011.

[30] R. B. Mansilha, L. Saino, M. P. Barcellos, M. Gallo, E. Leonardi, D. Perino, and D. Rossi, "Hierarchical Content Stores in High-Speed ICN Routers: Emulation and Prototype Implementation," in *Proc. ACM ICN*, 2015, pp. 59–68.

[31] G. Rossini, D. Rossi, M. Garetto, and E. Leonardi, "Multi-Terabyte and Multi-Gbps Information Centric Routers," in *Proc. IEEE INFOCOM*, 2014, pp. 181–189.