CoMon: An Architecture for Coordinated Caching and Cache-Aware Routing in CCN

Hani Salah* and Thorsten Strufe†
*TU Darmstadt, Germany, ⟨hsalah@cs.tu-darmstadt.de⟩
†TU Dresden, Germany, ⟨thorsten.strufe@tu-dresden.de⟩

***Abstract*—The autonomous cache management in Content-Centric Networking (CCN) results in suboptimal caching decisions and implies cache-ignorant routing. Cache coordination and similar improvements hence have been the subject of several recent studies. The proposed solutions, however, are either impractical due to their massive coordination overhead, or of limited benefit since they cannot realize perfect coordination.**

We present *CoMon*, an architecture for network-wide coordinated caching. CoMon realizes an affordable, yet highly effective, coordination by assigning monitoring and cache-aware (re)routing tasks to only a few nodes, through which the majority of traffic is expected or enforced to pass. CoMon, by design, maximizes the diversity of cached contents and reduces cache replacements. In addition, our simulation study using ISP topologies, shows that CoMon under several scenarios, when coordinates through a small ratio of the nodes, reduces the server hit ratio (i.e. the ratio of requests consumed by the origin content providers) of both CCN and notable related work, remarkably.

***Index Terms*—Information-Centric Networking; Coordinated Caching; Cache-Aware Routing**

I. Introduction

The Internet, which was designed for end-to-end communication, is dominated by content distribution and retrieval applications, today. This mismatch between the design and usage, beside the enormous (and ever-increasing [1]) volumes of traffic generated by content distribution applications, will soon disable the Internet from satisfying its users' demands. To cope with this situation, a broad research direction called Information-Centric Networking (ICN) proposed for a radical shift from the traditional host-centric communication model to an information-centric one. Notable ICN architectures include: Content-Centric Networking (CCN), 4WARD, DONA, and PSIRP (for an overview, please refer to [2]). A key feature of those architectures (that promises to improve the efficiency gains for Internet content distribution) is *in-network caching*, in which each node *autonomously* caches part of the contents passing through it, based on its local view of content access (i.e. names of requested contents and their properties, like frequency and recency of requests). In CCN [3], for instance, contents are cached at *all* intermediate nodes on the path from the content provider to the consumer (which is commonly known as *on-path caching*).

The autonomous cache management, although simple and does not cause coordination overhead, has three drawbacks. First, it is likely to cause unnecessary large cache redundancy (it thus poorly utilizes the available, already limited [4], cache space). Second, it may result in suboptimal selection for contents that are given priority for caching as the node's view of content access is likely different from the global (i.e. network-wide) view. Third, it implies cache-ignorant routing, and hence the system can take advantage of cached contents only when they are located *opportunistically* on the default path towards the origin content providers. These drawbacks have already raised doubts about achieving the intended caching efficiency of ICN, which match with prior results (e.g [5], [6]) showing that in some cases only a very small ratio of content requests hits a cache. Consequently, several studies (e.g. [6]–[9]) proposed to address these drawbacks through cache coordination.

We argue that coordinated cache management can be highly beneficial only when it is based on timely and network-wide knowledge of both content access information and cache configurations (i.e. which content is cached in which node?). The distributed nature of ICN beside the massive volumes and high dynamics of coordination information, however, render such coordination (hereafter, *perfect coordination*) impractical. Therefore, despite the considerable number of prior attempts to coordinate caching in ICN, they either favour low coordination cost over high performance gain or vice versa. To the best of our knowledge, a solution that realizes a close-to-perfect coordination with an affordable coordination overhead is still missing, and this is what we aim to achieve in this study.

Towards this end, we present and evaluate *CoMon* (our main contribution in this paper), an architecture for **Co**ordination that is based on *lightweight* **Mon**itoring of content access information and *bounded* advertisement of cache configurations. CoMon's design is guided by two requirements:

1) ***Realizing network-wide caching goals:*** CoMon should allow the network operator to apply network-wide goals (e.g. maximizing the diversity of cached contents in the *entire* network).
2) ***Incurring low coordination cost:*** The first requirement should be fulfilled with an affordable coordination overhead. We focus on the signalling overhead, which mainly relates to the number of coordinating nodes.

CoMon exploits the topological properties of the nodes to perform content monitoring and cache-aware (re)routing at only a few nodes, through which the majority of traffic is expected or enforced to pass. These nodes work with a controller who takes network-wide decisions in a central way. CoMon is consistent with CCN, the most studied ICN architecture. Its design concepts, however, are applicable for the other architectures. Our evaluations show that CoMon can remarkably reduces the server hit ratio of both the original CCN and prior work (particularly, [10] and [11]).

The rest of the paper is organized as follows. We give background information in Section II, motivate our study through an example in Section III, and discuss related work in Section IV. Next, we introduce CoMon in Section V, describe its specifications in Section VI, and evaluate it in Section VII. We finally conclude the paper in Section VIII.

II. Background

In this section, we first give an overview for CCN in Section II-A, before introducing the main terms and performance metrics that we use along the paper in Section II-B and Section II-C, respectively.

A. Primer on CCN

Content-Centric Networking (CCN) [3] is a network architecture for future Internet proposed by Van Jacobson, aiming to coincide the Internet's design and its usage (which is currently dominated by content distribution traffic [1]). CCN adopts three core concepts:

1) ***Networking named content:*** Contents are identified and addressed by unique hierarchical names (rather than locations or host addresses).
2) ***On-path caching:*** Contents are cached along the delivery path from the content provider to the consumer.
3) ***Consumer-driven communication model:*** Named content is first requested by an *Interest* packet, and then the content itself is delivered inside a *Data* packet on the same path (in reverse way).

The node model in CCN consists of three data structures: the *Content Store (CS)* (or *cache*) holds Data packets passing through the node. The *Pending Interest Table (PIT)* maintains content names of recently received, but still not consumed (i.e. satisfied), Interest packets along with the interfaces (or *faces*) through which they were received. The *Forwarding Information Base (FIB)* maintains a list of potential next-hop interfaces for different content names and prefixes. With this node model, CCN nodes handle packets as follows: upon receiving an Interest packet, the node first looks for a matching name in its CS; if found, it forwards the corresponding Data packet to the same interface from which the Interest was received. Otherwise, the node looks for the name in its PIT; if a matching entry is found but the interface from which it was received is not listed, the new interface is appended to the same entry, and nothing otherwise. This way, the nodes avoid forwarding duplicate copies of identical Interest packets. If no matching PIT entry found, a new PIT entry is created, then the FIB is consulted, and the packet is forwarded accordingly.

When receiving a Data packet, the node first looks for the content name in its PIT; if found, the Data packet is cached in the CS (LRU or LFU is used for content replacement, in case the CS is full) and then forwarded to all the listed interfaces, and lastly the respective PIT entry is deleted. If no matching PIT entry found, the packet will be discarded.

B. Terminology

We summarize here the terminology that we use along the paper: *uncoordinated or original CCN* refers to CCN as described in [3] (or Section II-A). *Coordinated CCN* refers to a CCN-like system whose nodes (or part of them) exchange their content access views and/or cache configurations to coordinate their caching-related decisions. If cache configurations are exchanged, the coordination implicitly enables for *cache-aware routing*. *Perfect coordination* implies that *all* network nodes exchange *timely* information of *both* their content access views and cache configurations.

C. Performance Metrics

In this paper, we use the following, commonly used, metrics to evaluate the performance of CCN-like systems:

1) ***Cache diversity:*** The ratio of unique contents in the caches (to the overall cache capacity of the network).
2) ***Server hit ratio:*** The ratio of the Interest packets consumed by the origin content providers.
3) ***Path stretch:*** The number of hops that the Data packet has traversed, normalized over the number of hops between the consumer and the origin content provider.
4) ***Content replacements:*** The number of content replacements occurred. More replacements cause more memory access and processing power from the caching nodes.

III. Illustrative Example

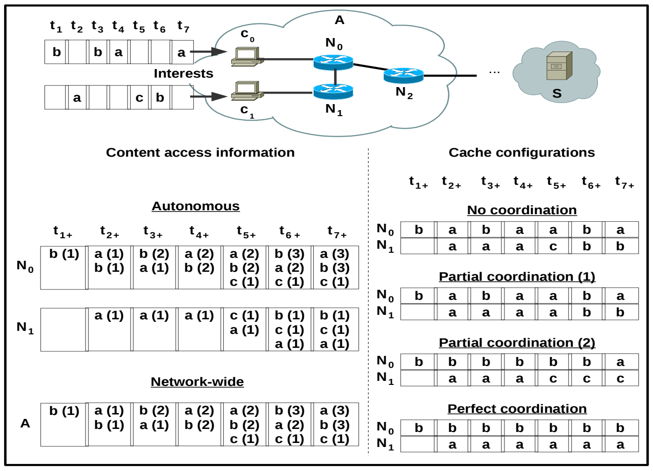
This example illustrates both the benefits and challenges of coordinating caching decisions in CCN [1]. As shown in Fig. 1, there is a network A and a scenario consisting of seven Interest packets. t_i denotes the time-stamp at which each Interest packet was issued. The network A comprises two ingress nodes N_0, N_1, and one egress node N_2 (connecting A to the rest of the Internet). Each of N_0 and N_1 has a cache store (CS) that is capable of holding one content, whereas N_2 has no CS. N_0 and N_1 use LFU for cache replacements (LRU is used in case of equality). c_0 and c_1 abstract clients connecting to N_0 and N_1, respectively. The server S abstracts all origin content providers. We assume that $distance(N_2, S) = 10$ hops.

The example compares the performance of four different systems: a system without coordination, a system applying perfect coordination (as defined in Section II-B), and two systems applying partial coordination. In both the two systems with partial coordination, we assume that all the nodes participate in coordination; they differ in the information the nodes exchange among each others for coordination. In the first (hereafter, *partial-1*), the nodes exchange only content access information, while in the second (hereafter, *partial-2*) they exchange only cache configurations.

[1]This is an extension for an example from [12]. More precisely, our example considers more aspects (covering all the cases that relate to our study), using a more comprehensive scenario and more evaluation metrics.

TABLE I: Performance Results of the Illustrative Example

	No coordination	**Partial coordination (1)**	**Partial coordination (2)**	**Perfect coordination**
Cache diversity	(1+0.5+1+0.5+1)/5 = 80%	(1+0.5+0.5+0.5+1)/5 = 70%	(1+1+1+1+1)/5 = 100%	(1+1+1+1+1)/5 = 100%
Server hit ratio	(1+1+1+1+1)/5 = 100%	(1+1+1+1+1)/5 = 100%	(0+0+1+0+1)/5 = 40%	(0+0+1+0+0)/5 = 20%
Path stretch	(12+12+13+13+12)/62 = 100% [2]	(12+12+13+13+12)/62 = 100%	(1+2+13+2+12)/62 $\approx$ 48%	(1+2+13+2+2)/62 $\approx$ 32%
Replac. / node	(1+1+0+1+1+1+1)/7 $\approx$ 86%	(1+1+0+0+1+1+1)/7 $\approx$ 71%	(0+0+0+1+0+0+1)/7 $\approx$ 29%	(0+0+0+0+0+0+0)/7 = 0



t1 t2 t3 t4 t5 t6 t7
b b a a
a c b
Interests
A
c0
c1
N0
N1
N2
...
S
Content access information
Autonomous
t1+ t2+ t3+ t4+ t5+ t6+ t7+
N0
b (1) | a (1) b (1) | b (2) a (1) | a (2) b (2) | a (2) b (2) c (1) | b (3) a (2) c (1) | a (3) b (3) c (1)
N1
a (1) | a (1) | a (1) | c (1) a (1) | b (1) c (1) a (1) | b (1) c (1) a (1)
Network-wide
A
b (1) | a (1) b (1) | b (2) a (1) | a (2) b (2) | a (2) b (2) c (1) | b (3) a (2) c (1) | a (3) b (3) c (1)
Cache configurations
t1+ t2+ t3+ t4+ t5+ t6+ t7+
No coordination
N0 b a b a a b a
N1 a a a c b b
Partial coordination (1)
N0 b a b a a b a
N1 a a a a b b
Partial coordination (2)
N0 b b b b b b a
N1 a a a c c c
Perfect coordination
N0 b b b b b b b
N1 a a a a a a

Fig. 1: Topology and Scenario of the Illustrative Example

In both the uncoordinated system and partial-1, Interest packets are forwarded along the shortest path towards S, and consumed either by S or opportunistically by a cache on the path. As for the other two systems, Interest packets are forwarded either towards the other caching node (if the content is cached there), or towards S otherwise.

Fig. 1 shows, at each time-stamp t_{i+} (i.e. after t_i and before t_{i+1}), the content cached at each of N_0 and N_1, in each of the four systems. In both the uncoordinated system and partial-1, each node ranks the observed contents, according to its knowledge of content access, and caches the top ranked one. In the other two systems, the network-wide goal is to diversify cached contents; that is, each node ranks the observed contents, according to its knowledge of content access and global knowledge of cache configurations, and caches the top ranked content unless it is cached by the other caching node; in that case, it caches the second ranked content.

The results, based on four commonly used performance metrics (Section II-C), are summarized in Table I, comparing the four systems from t_3 onwards (i.e. after all the caches became full). As to be expected, the results show that the system with full coordination outperforms the others (for all the four metrics), and that the two systems with partial coordination (particularly partial2) outperform the system without coordination[3]. However, this superiority comes with price (i.e. the *coordination cost*), represented by the volume of traffic that need to be exchanged for coordination between N_0 and N_1.

[2] The denominator (i.e. no. of hops till S) = 12+12+13+13+12 = 62.

[3]Note that N_0's local view of content access without coordination is the same as with perfect coordination. This is attributed to N_0's central position, and because none of the Interest packets issued by c_1 were consumed by N_1's cache. Nevertheless, this knowledge *alone* was not sufficient for N_0 to take decisions as efficient as if the cache configuration of N_1 was also known.

IV. Related Work

Existing solutions for coordinated caching in ICN can be classified as either *indirect* (*implicit*) or *direct* (*explicit*). Both classes aim to outperform uncoordinated CCN (in terms of one or more of the metrics listed in Section II-C). With indirect coordination, nodes do not need to exchange coordination information among each others (or exchange little information). Instead, each node takes decisions independently according to a predefined strategy (allowing for implicit cooperation), based only on its local content access view. Notable examples include: [6], [13], Leave Copy Down (LCD) [11], and probabilistic caching [10]. In the solutions that apply direct coordination (e.g. [7]–[9]), in contrast, the nodes take caching-related decisions based on the information they explicitly exchange among each others. This way, solutions with indirect coordination generally favour low coordination cost over high performance gain, while direct coordination solutions make the opposite. Due to space constraints, we do not describe each of those solutions individually; recent survey [2], [14] give more details than we can provide here.

To the best of our knowledge, among existing ICN coordination solutions, only the designs of [8] and [9] may realize almost perfect coordination (as defined in Section II-B). To do so, either all the network nodes [8] or all ingress nodes [9] should participate in the coordination. The high coordination overhead of those solutions, however, likely makes them impractical. Therefore, each of the two papers proposed a *lighter* version of its original solution. As for the other solutions, although their evaluations show that they outperform the uncoordinated CCN with relatively low coordination cost, their gains are still limited compared to what can be achieved with perfect coordination. This is obvious, for instance, when comparing the performance of the original coordination solutions of [8] and [9] to their respective lighter versions.

That said, as yet there is no solution that can realize (almost) perfect coordination with an affordable cost. This is the goal that CoMon (our solution for coordination) aims to achieve. While a preliminary version of this work appeared in [15], this paper describes CoMon's design in detail and evaluates it through extensive simulations with realistic settings.

V. CoMon: A High-Level Overview

Coordination can be done either within a domain or can involve multiple autonomous systems. The CoMon version that we present in this paper is designed to work in a domain-wide scale. Each domain (or network) consists of a set V of nodes (with cardinality $|V|$). We assume homogeneous cache capacities for all the nodes. We let c and C denote the cache capacity per node and the overall cache capacity of the entire domain, respectively (i.e. $C = c.|V|$).

Fig. 2 shows the system architecture of CoMon. It consists of the following three components:

1) ***Caching Controller (CC):*** Each domain has a controller that: (*i*) aggregates monitored content access information, (*ii*) calculates and commands caching-related decisions, and (*iii*) advertises cache configurations in the routing plane of only a set $M \subset V$ of nodes. Motivated by prior results (e.g. [16], [17]), showing that centralized management can significantly decrease coordination cost and complexity, we chose to implement a centralized CC.
2) ***Caching Nodes (CNs):*** These are similar to CCN nodes [3] in that they have routing and caching capabilities. However, instead of taking autonomous decisions, each CN caches and evicts contents as commanded by the CC. In response to a command implying to cache contents, the node: (*i*) identifies which of these contents are not already in its cache, (*ii*) requests each of them by an Interest packet (i.e. *prefetching*), and creates a respective PIT entry and initializes it with a null interface, and (*iii*) stores the respective Data packets when arrive after verifying their integrity and provenance (to avoid cache poisoning). The CN appends the interface information of matching Interest packets that arrive before the CN receives the corresponding Data packets to the respective PIT entries.
3) ***Caching Nodes with Monitoring and Re-routing capabilities (CNMRs):*** A set of $M \subset V$ nodes are selected as CNMRs. In addition to a CN's regular functions[4], each CNMR: (*i*) monitors each Interest packet passing through it, and records its content's names and characteristics (e.g. number of times it has been requested during the current observation period), (*ii*) periodically (as determined by the CC) uploads the recorded content information (hereafter, *monitoring reports*) to the CC, and clears them afterwards, and (*iii*) receives lists of cached contents from the CC, and (re)routes Interest packets accordingly.

Please note that the system operation described above implies *off-path* caching and supports cache-aware routing. It also gets rid of CCN's requirements of performing cache replacements and signature verifications at line rate.

VI. CoMon: Design Specifications

In this section, we describe the design specifications of CoMon that are not detailed in Section V.

[4]From here ahead, we use the term *node* in a generic way for referring to any node $v \in V$ (i.e. either CN or CNMR), and use *CN* for referring to a node u without monitoring nor re-routing capabilities (i.e. $u \in V : u \notin M$).

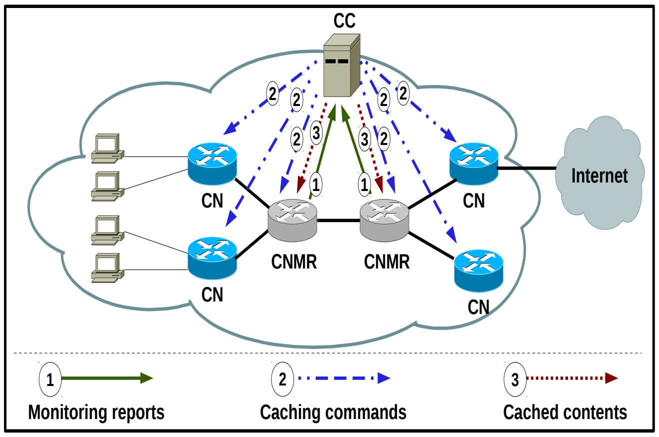


Fig. 2: CoMon Architecture (CC: Caching Controller; CN: Caching Node; CNMR: Caching Node with Monitoring and Re-routing capabilities). The coordination messages are unsolicited Data packets, belonging to a reserved name prefix; it can be determined whether they are legitimate or not by verifying the contained signatures.

A. Caching-Related Decisions

After each observation period (i.e. after receiving new monitoring reports from the CNMRs), the CC calculates caching-related decisions according to the network-wide goals specified by the network operator. In particular, the CC has to determine: (*i*) the contents that are given priority for caching, and (*ii*) their positions in the network. Those decisions implicitly determine the redundancy degree for each selected content as well as its caching duration (each cached copy is preserved till a more recent decision implies that it has to be evicted). The above-described system operation implies that the CC has a full control over the entire caching process, which makes CoMon compatible with *any* network-wide goals.[5]

Without losing the generality, we experiment CoMon in this paper with an *exemplary* network-wide policy: Maximizing the diversity of popular contents cached inside the network, thus favouring data availability over the hop count required for data delivery inside the network. This policy was also adopted in several studies (e.g. [6], [9]), and it mainly aims to minimize the network traffic that crosses other domains because the transit cost is commonly charged according to the traffic volume [19]. In the following, we describe how the CC implements the aforementioned two decisions accordingly:

1) ***Content Selection for Caching****:* The CC selects contents that should be given priority for in-network caching by maintaining a list containing the names of both the currently cached contents and the recently monitored ones. It calculates the *popularity score* f_i for each content i, and ranks the contents accordingly. Given our exemplary goal to maximize the diversity of cached contents, the top C ranked contents are selected for caching (C denote the overall cache capacity). In our experiments (Section VII), we use the LFU algorithm for calculating the popularity scores.

[5]Please note that this design allows the CC to mitigate cache pollution attacks at a network-wide scale, which we expect to be more effective than prior node-level (i.e. autonomous) mechanisms like [18]. We leave the design and evaluation of such *add-on* mechanism for future work.

*2) **Content Placement (Content-to-Cache Distribution)**:* Assuming that both content popularity and nodes' betweenness centrality (BC) scores have power-law distributions, Wang et al. [20] observed that, with our approach for content selection (Section VI-A1) and the exemplary network-wide goal of maximizing content diversity, a high network performance is achieved when the contents with highest popularity are placed in the network core (i.e. on the nodes that have highest BC scores) while the less popular ones are placed at the edges.

Motivated by the aforementioned observation, CoMon places contents in the network as follows: First, the BC score for each node $v \in V$ (counting the total fraction of all shortest paths in the network that pass through v [21]) is calculated. Second, the nodes are sorted by their BC scores in descending order. Third, the selected contents, from the most to least popular, are assigned iteratively on the sorted nodes.

B. Reducing the Frequency of Changing Cache Configurations

Changing cache configurations after each observation period, according to the newly calculated caching decisions, results in: (*i*) signalling overhead for delivering both the caching assignments (from the CC to the nodes) and the list of newly selected contents (from the CC to the CNMRs), beside (*ii*) computational and memory access overhead both for updating the CNMRs' FIBs and for content replacements.

In order to mitigate this overhead, CoMon implements a technique in which the CC first calculates the *feasibility of changing* (FoC) the set H_c of currently cached contents to the set H_n of newly selected contents:

$$FoC = 1 - \frac{\sum\limits_{x \in H_c, x \in H_n} f_x}{\sum\limits_{y \in H_n} f_y} \tag{1}$$

where f_x and f_y denote the newly calculated popularity scores of contents x and y, respectively. The CC then changes the current configurations only if the resulted FoC value was above a preset threshold.

C. Cache-Aware Routing (CAR)

CoMon implements CAR as follows: each Interest packet is forwarded along the shortest path towards the corresponding origin content provider till it encounters a CNMR. Only that (i.e. first encountered) CNMR checks whether the requested content is among the currently cached contents or not (according to the list it received from the CC). If not, the original path is preserved; otherwise, the CNMR adds the ID of the matching node (node IDs are treated as content names; hereafter, *node names*) as a name prefix to the original content name, and reroutes the packet towards the identified node name. For example, if the content ***"/org/ieee-ccnc/papers/comon.pdf"*** is cached at N_7, the CNMR changes the name to: ***"/N7/org/ieee-ccnc/papers/comon.pdf"***. In either case, the CNMR before forwarding the packet, sets a *"checked"* flag, so that next encountered CNMRs skip the aforementioned checking step.

This procedure requires that each node stores *additional* $|V| - 1$ FIB entries, one towards each of the other nodes in the domain, so that all the nodes can route packets toward each others.

D. Selection of CNMRs

Intercepting all or the majority of traffic by CNMRs is essential for two reasons: First, it enables the CNMRs to generate accurate monitoring reports (which increases the quality of the consequent caching decisions). Second, it increases the chance to reroute Interest packets to local caches.

The corresponding *CNMR selection problem* can be expressed as follows: *"Given a domain consisting of a set V of nodes, which set $M \subset V$ of nodes should be selected as CNMRs such that their aggregate traffic coverage is maximized while $|M|$ is minimized?"*. This problem is known to be NP-hard [22], and we hence aim at a heuristic selection.

Among existing solutions (for an overview, please refer to [22], [23]), we chose the *group betweenness centrality (GBC)* [24], which counts the total fraction of all shortest paths that pass through at least one node in a given group of nodes. That is, the $|M|$ nodes with the highest GBC score are selected as CNMRs. GBC is an extension for the well-know betweenness centrality (BC) metric [21], which is considered to be highly correlated with the network traffic [25] (particularly in scale-free networks, like ISP topologies and the Internet). GBC, however, was shown to be more effective than using the same number of nodes with highest BC scores [26].

While these features are appealing, GBC does not *guarantee* a complete or very high traffic coverage. The coverage could be even worse in CCN-like networks since caches and PITs may filter part of the traffic before it is intercepted by a CNMR. In order to improve the traffic coverage achieved by the above-described GBC-based placement, CoMon incorporates the following two techniques; when both work jointly, they *enforce* each Interest packet to pass through a CNMR.

*1) **Monitor-Aware Routing (MAR)**:* Ignoring the filtering effects of caches and PITs for now, MAR modifies the original routing strategy such that each Interest packet crosses at least one CNMR. We propose two versions of MAR (hereafter, *MAR-V1* and *MAR-V2*). Both versions involve two routing steps (only the first step is different). In MAR-V1, the ingress node first routes the packet towards its closest CNMR (after adding the node name of the designated CNMR as a prefix for the original content name). Next, the designated CNMR removes the added name prefix, then routes the packet either towards the matching local node if the content is cached, or towards the respective origin content provider otherwise.

It is obvious that, unless the default path contains a CNMR, MAR results in additional routing hops. MAR-V2 aims to minimize this overhead by selecting the destination CNMR in the first routing step (i.e. at the ingress node) according to the requested content name. This is done using the *FINDING-FIRST-CNMR Procedure* (Algorithm 1), which is adapted from the *Routing-with-Content-Filtering (RFC) algorithm* [27]. The ingress node passes two parameters to the procedure: (*i*) its own name, and (*ii*) the name of the requested content.

In Algorithm 1, the procedure (lines: 5 – 13) compares the costs of all the matching paths (i.e. from the ingress node towards the egress node through which the origin content provider is accessed[6]) that contain CNMR(s); the total path cost (line 8) sums up two values: the cost from the ingress node to the CNMR (line 6) and the cost from the CNMR to the egress node (line 7). The procedure then selects the lowest cost path (lines: 9 – 12), and among the CNMRs locating on it, the name of the ingress node's closest CNMR is returned (line 14). The ingress node in turn updates its FIB accordingly, so it can route upcoming Interest packets towards the same content name without recalculating the destination CNMR again.

Note that, compared to non-MAR CoMon, MAR-V1 reduces the FIB overhead of CNs to $\mathcal{O}(|V|-1)$, since each CN needs only to reach the other $|V|-1$ nodes; one of them is labelled as its closest CNMR. As for CNMRs, their FIB overhead is the same as with non-MAR CoMon; the same applies to the FIB overhead both for CNs and for CNMRs when MAR-V2 is applied.

Algorithm 1 Finding the Best Route's First CNMR

```
1:  // Find the best route from an ingress node $s$ towards content's
    corresponding egress node $t$ that crosses CNMR(s). Returns the
    CNMR $m_c$ (closest to $s$)
2:  procedure FINDING-FIRST-CNMR($s, t$)
3:      // Initialization
4:      $min\_sum \leftarrow largest\_spc$          ▷ spc: shortest path cost
5:      for each CNMR $m$ in $M$ do
6:          compute $spc(s, m)$
7:          compute $spc(m, t)$
8:          $sum = spc(s, m) + spc(m, t)$
9:          if $sum < min\_sum$ then
10:             $min\_sum \leftarrow sum$
11:             $m_c \leftarrow m$
12:         end if
13:     end for
14:     Return $m_c$
15: end procedure
```

*2) **Forward-Till-Be-Monitored (FTBM):*** As mentioned earlier, Interest packets in CoMon (as in CCN) may be consumed by a cache or captured by a PIT before they encounter a CNMR. To mitigate the effects of these filters, CoMon implements FTBM, which works as follows: when a node receives an Interest packet and finds a matching Data packets in its cache or a matching PIT entry, the node adds a preserved name prefix ***"/served/"*** to the original name, and then forwards the packet to the closest CNMR. That CNMR, in turn, extracts the original content name, updates the monitoring information accordingly, and drops the packet afterwards.

The additional overhead caused by FTBM is measured by the number of hops traversed by the Interest packet after it has been served till it reaches a CNMR. Please note that since the Interest is already consumed, this overhead does not apply for the respective (relatively much larger) Data packet.

VII. EVALUATION

We performed a simulation study to evaluate both the reduction in server hit ratio (SHR) that can be achieved with CoMon (Section VII-B), and the hop count overhead of MAR (Section VII-C). As for the other metrics listed in Section II-C, CoMon by design (Section VI) can maximize the cache diversity and reduces cache replacements. Before discussing the results, we describe our simulation setup in Section VII-A. All results represent the respective systems after stabilization.[7]

A. Simulation Setup

We used ccnSim [28], a highly scalable CCN simulator. We adapted ccnSim in order to produce the behaviour of CoMon. We have also simulated the original CCN and two notable coordination solutions: (*i*) ProbeCache [10] in which contents are cached at each node with a probability determined by the node's distance from the requester (the closer the node, the higher the caching probability); (*ii*) Leave Copy Down (LCD) [11] in which a cache hit results in caching the content at the direct downstream node. LRU was used for cache replacement in CCN, ProbeCache, and LCD simulations.

We fed the simulator with real ISP topologies measured by the Rocketfuel project [29]. We experimented with four different topologies. However, since the conclusions equally apply for all the topologies, we discuss the results of one topology only: the Exodus ISP (AS 3967) topology, consisting of 79 nodes and 147 bidirectional edges.

Each experiment was repeated 20 times. At the beginning of a run, the simulator *randomly* picks $\lceil 70\% \rceil$ of the network nodes as ingress nodes, and three of the rest as egress nodes. In each run, $1,000,000$ content requests (i.e. Interest packets) in total are generated from all clients modelled as Poisson processes. Contents are permanently stored at servers located outside the ISP. Each content is accessible via only one egress router, selected randomly at the beginning of the run.

The simulations applied a high competition among the contents on the available storage by using small cache sizes, with a homogeneous size $c = 10$, and a relatively large content population $P = 100.C = 79,000$. Similar to most of related work, following measurement results from the real Internet, content popularity has been modelled as a Zipf distribution: we performed simulations under varying content distribution skewness (i.e. Zipf parameter) $\alpha \in \{0.8, 1.2, 1.5\}$. In addition, to catch a very bad case of unpopular contents, we have also simulated an additional case modelling content popularity by the Mandelbrot-Zipf (MZipf) distribution using the parameters: $\alpha = 0.8$ and $q = 5$.

With CoMon, we set FoC threshold to 0.2. As for CNMRs, we experimented with two settings: (*i*) with the top $\lceil 5\% \rceil$ GBC nodes, and (*ii*) with all ingress routers (as in [9]). As to be

[6]We assume that each content's origin provider is accessible through one specific egress node only. However, extending Algorithm 1 such that to consider the multi-egress case is straightforward.

[7]For details, please refer to: http://www.infres.enst.fr/ drossi/ccnSim/ccnsim-manual-0.3.pdf.

expected, changing the ratio of CNMRs did not affect the traffic coverage as long as both MAR and FTBM are enabled, thus the server hit ratio results were also almost identical; we therefore show below only the results of the first setting. Please note that the values of the first and third parts of the signalling overhead (messages labelled (1) and (3) in Fig. 2) with the first setting represent only about $\frac{\lceil 5\% \rceil}{\lceil 70\% \rceil}$ the respective values with the second setting.

B. Server Hit Ratio

Fig. 3 depicts the server hit ratio (SHR) results for the different systems. CoMon was simulated both with MAR-V1 and MAR-V2, with FTBM enabled. The SHR results of both MAR-V1 and MAR-V2, as to be expected (since both apply same level of coordination based on same type and amount of information), were almost identical, and therefore we only show the results with MAR-V1. All in all, the results show that CoMon remarkably reduces the SHR of the other systems. In particular, CoMon reduces the SHR of LCD (the best among the three other systems) by about 30% for $\alpha = 0.8$. Although the SHR results of all the four systems improve (i.e. decrease) for larger α, [8] CoMon was always the superior with a notable difference.

This superiority of CoMon confirms the utility of coordination based on global knowledge of content access information and cache configurations. This knowledge enabled CoMon to produce efficient caching-related decisions, which resulted in maximizing the ratio of Interest packets that are consumed by intra-domain caches.

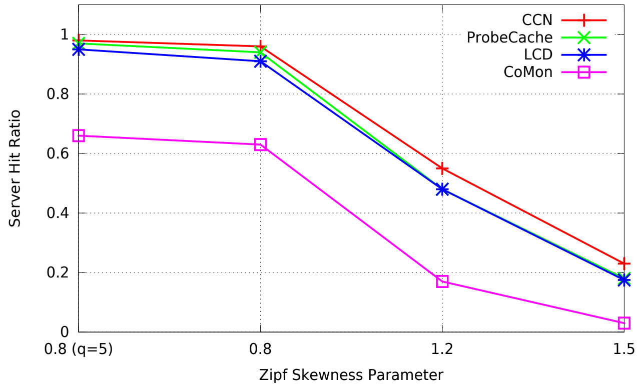


Fig. 3: Server Hit Ratios of Different CCN-like Systems

C. The Hop Count Overhead of MAR and FTBM

We aim here to measure the hop count overhead of both MAR-V1 and MAR-V2, compared to the hop count in the original CCN. To measure this overhead, we focus on hop counts of the Interest packets which were consumed by origin content providers. Fig. 4 plots the CDF of the hop count distribution, counting the hops from the ingress node till the egress node. The results, in general, show that the hop count overhead of MAR is relatively small, but (as to be expected) is slightly better with MAR-V2. For instance, while about half the packets traversed a maximum of 4 hops without MAR, this was achieved by about 31% and 35% of the packets with MAR-V1 and MAR-V2, respectively. We consequently argue that the positive impact of MAR on SHR (Section VII-B) pays off the consequent hop count overhead.

[8]Similar results have been observed for CCN, ProbeCache, and LCD previously (please refer to [5] for explanation).

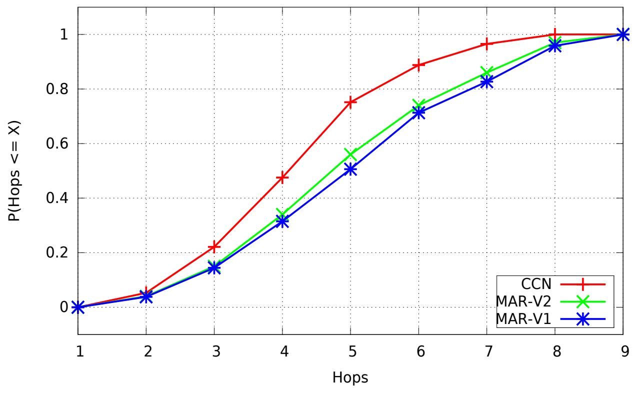


Fig. 4: CDF of Hop Count Distributions: MAR-V1 vs. MAR-V2 vs. CCN (MZipf: $\alpha = 0.8, q = 5$)

VIII. Conclusion

We presented a CCN-like architecture called CoMon. It realizes a reasonably priced, yet close-to-perfect, coordination, and enables cache-aware routing, thus achieves an outstanding performance. Table II summarizes which of CoMon's techniques enables to achieve which of the preset design requirements.

TABLE II: Mapping CoMon's Design Requirements and Design Concepts

Realizing network-wide goals	**Incurring low coordination cost**
The CC has a centralized control over the entire caching process.	Employing few CNMRs with a centralized control.
MAR and FTBM allow to gather global content knowledge.	Each Interest packet is checked only once.
CAR allows to fully leverage cached contents.	The FoC technique reduces the frequency of updates.

Our vision for CoMon is broader than its use for cache coordination only. We are studying the extension of CoMon in several directions, e.g. as an infrastructure for network statistics, for the detection and mitigation of several attacks, as well as in solving content's versioning and consistency problems. We are also considering the use of CoMon in a multi-AS scenario where ASes maintain content-level peering agreements to leverage each others' cached contents.

References

[1] "Cisco visual networking index: Forecast and methodology, 2013–2018," *CISCO White paper*, 2014.
[2] G. Xylomenos *et al.*, "A survey of information-centric networking research," *IEEE Communication Magazine*, 2013.
[3] V. Jacobson *et al.*, "Networking named content," in *CoNEXT*, 2009.
[4] D. Perino and M. Varvello, "A reality check for content centric networking," in *SIGCOMM ICN workshop*, 2011.

[5] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep., 2011.
[6] S. Saha, A. Lukyanenko, and A. Yl-Jski, "Cooperative caching through routing control in information-centric networks," in *INFOCOM*, 2013.
[7] Y. Wang *et al.*, "Advertising cached contents in the control plane: Necessity and feasibility," in *INFOCOM Workshops*, 2012.
[8] V. Sourlas, P. Flegkas, and L. Tassiulas, "Cache-aware routing in information-centric networks," in *IFIP IM*, 2013.
[9] D. Saucez *et al.*, "Minimizing bandwidth on peering links with deflection in named data networking," in *ICCIT*, 2013.
[10] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *SIGCOMM ICN workshop*, 2012.
[11] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Elsevier Performance Evaluation*, 2006.
[12] Y. Li *et al.*, "Coordinating in-network caching in content-centric networks: model and analysis," in *ICDCS*, 2013.
[13] W. K. Chai *et al.*, "Cache less for more in information-centric networks," in *NETWORKING*, 2012.
[14] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: a survey," *Elsevier Computer Networks*, 2013.
[15] H. Salah, B. Schiller, and T. Strufe, "Comon: A system architecture for improving caching in ccn," in *INFOCOM Student Workshop*, 2014.
[16] A. Greenberg *et al.*, "A clean slate 4D approach to network control and management," *SIGCOMM CCR*, 2005.
[17] V. Sekar *et al.*, "cSamp: A System for Network-Wide Flow Monitoring," in *NSDI*, 2008.
[18] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *INFOCOM*, 2012.
[19] S. DiBenedetto, C. Papadopoulos, and D. Massey, "Routing policies in named data networking," in *SIGCOMM ICN workshop*, 2011.
[20] L. Wang, S. Bayhan, and J. Kangasharju, "Effects of cooperation policy and network topology on performance of in-network caching," *IEEE Communications Letters*, 2014.
[21] L. C. Freeman, "A set of measures of centrality based on betweenness," *JSTOR Sociometry*, 1977.
[22] K. Suh *et al.*, "Locating network monitors: complexity, heuristics, and coverage," *Elsevier Computer Communications*, 2006.
[23] N. Zhu *et al.*, "Overview of monitor selection in computer networks," in *Springer Trustworthy Computing and Services*, 2013.
[24] M. G. Everett and S. P. Borgatti, "The centrality of groups and classes," *Taylor & Francis - The Journal of mathematical sociology*, 1999.
[25] P. Holme, "Congestion and centrality in traffic flow on complex networks," *Advances in Complex Systems*, 2003.
[26] R. Puzis, Y. Elovici, and S. Dolev, "Fast algorithm for successive computation of group betweenness centrality," *Physical Review E*, 2007.
[27] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Configuring networks with content filtering nodes with applications to network security," in *INFOCOM*, 2005.
[28] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnsim: An highly scalable ccn simulator," in *ICC*, 2013.
[29] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *SIGCOMM*, 2002.