A Methodology for the Design of Self-Optimizing, Decentralized Content-Caching Strategies

Karla Kvaternik, Jaime Llorca, Daniel Kilper, *Senior Member, IEEE*, and Lacra Pavel, *Senior Member, IEEE*

***Abstract*—We consider the problem of efficient content delivery over networks in which individual nodes are equipped with content caching capabilities. We present a flexible methodology for the design of cooperative, decentralized caching strategies that can adapt to real-time changes in regional content popularity. This design methodology makes use of a recently proposed *reduced consensus optimization* scheme, in which a number of networked agents cooperate in locating the optimum of the sum of their individual, privately known objective functions. The outcome of the design is a set of dynamic update rules that stipulate how much and which portions of each content piece an individual network node ought to cache. In implementing these update rules, the nodes achieve a collectively optimal caching configuration through nearest-neighbor interactions and measurements of local content request rates only. Moreover, individual nodes need not be aware of the overall network topology or how many other nodes are on the network. The desired caching behavior is encoded in the design of individual nodes' costs and can incorporate a variety of network performance criteria. Using the proposed methodology, we develop a set of content-caching update rules designed to minimize the energy consumption of the network as a whole by dynamically trading off transport and caching energy costs in response to changes in content demand.**

***Index Terms*—Adaptive algorithms, algorithm design, content distribution, content-centric networking, decentralized coordination control, distributed algorithms, distributed content caching, distributed optimization, energy-efficient ICTs, intelligent networks, meshed networks, multiagent systems, nearest-neighbor interactions, reduced consensus optimization, self-optimizing networks.**

I. Introduction

A SIGNIFICANT portion of today's Internet traffic involves the delivery of content such as video to a multitude of geographically distributed users who are typically indifferent to where that content is stored or accessed from. According to CISCO's 2012 VNI report [1], video streaming and downloads presently account for over 86% of all Internet traffic, and services such as IPTV and Video-on-Demand (VoD) constitute the fastest growing Internet service class. Current trends toward ubiquitous mobile computing suggest that Internet traffic will continue to be dominated by the distributed on-demand consumption of video content.

The problem of content dissemination to a distributed set of users is ideally addressed by some form of multicasting, whereby each piece of data is delivered through a single transmission from the source, and copies are created only at branch points in the distribution tree.

Multicast technologies such as IP and application-layer multicast are well suited for the delivery of real-time multimedia services such as IPTV and live video streaming, in which requests for the same data are served simultaneously. However, the efficiency of multicasting cannot be directly exploited for the delivery of on-demand services such as VoD and time-shifted TV, in which content requests arrive asynchronously. When the same piece of data is accessed at different times from multiple locations, caching the data temporarily at intermediate nodes can enable multicast delivery, significantly reducing load at origin servers, access latency, and network congestion [2]. It is therefore widely accepted that caching is essential to enabling on-demand content access [2]–[7].

Several caching-based content distribution technologies are already extensively deployed, including privately owned content delivery networks (CDNs) such as Akamai and Limelight, peer-to-peer (P2P) systems such as BitTorrent, and Web caching solutions such as Squid and NetCache. In all cases, the central idea is that replicating content and caching it throughout the network facilitates the realization of important performance benefits such as reduced network congestion and server loading, reduced access latencies, and improved tolerance to transport disruptions.

However, these technologies are typically incompatible with one another, and they are overlay solutions implemented atop a host-centric Internet that was never intended for the mass distribution of content. As such, they often result in the suboptimal utilization of network resources [8], [9].

The Internet's fundamental incompatibility with today's content consumption trends has therefore prompted research into a new networking paradigm known as *information-centric networking* (ICN) [6], [10], in which content is directly accessed by name, rather than the address of its host. In an information-centric network, content can be delivered from any network location that caches a valid copy of that content, data packets can be transparently cached as they travel toward their consumers, and content can be assembled at its destination from data packets that may arrive from multiple locations. Compared to existing overlay solutions, ICN architectures are therefore more naturally poised to leverage in-network caching [6], [11].

Manuscript received April 24, 2013; revised June 06, 2014; December 24, 2014; and May 29, 2015; accepted August 19, 2015; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. X. Liu. Date of publication October 08, 2015; date of current version October 13, 2016. This work was supported by the NSERC under the Vanier fund and the NSERC CRD and Bell Labs/Alcatel-Lucent.

K. Kvaternik and L. Pavel are with the Edward S. Rogers Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: kvaternik@utoronto.ca; pavel@utoronto.ca).

J. Llorca and D. Kilper are with Bell Labs, Alcatel-Lucent, Holmdel, NJ 07733 USA (e-mail: jaime.llorca@alcatel-lucent.com; dan.kilper@alcatel-lucent.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2478059

The use of in-network caching has also been investigated as a possible means of reducing a network's energy consumption [12]–[15]. Although information and communication technologies (ICTs) are currently estimated to account for only 2% of the world's total carbon footprint, this proportion is expected to grow as ICT energy efficiency improvements plateau due to fundamental theoretical and physical limitations [16]. The aggressive growth in the number of users and the variety of demanded services motivate research into new ways of reducing the energy consumption of ICTs.

Regardless of the adopted technology (i.e., P2P, CDNs, or ICN), the extent to which the benefits of in-network caching can be realized depends crucially on the efficacy of the implemented *content caching strategy* (CCS). A CCS is a set of policies or algorithms that prescribe how much of what content each participating network node ought to cache. These may include various file placement and eviction policies, as well as protocols that ensure the consistency of content replicas [17].

An "effective" CCS is one that is *decentralized*, *adaptive* to real-time changes in regional content demand patterns, and allows individual nodes to be *nescient* with regard to the operation of the collective.

In a decentralized CCS, network nodes collectively achieve a desirable caching configuration by individually making independent caching decisions based on local network measurements and interactions with their nearest neighbors. The difficulties associated with implementing centralized coordination schemes in large networks obviate the need for decentralization; the acquisition of network measurements by a central node and the subsequent dissemination of coordination signals to each node consume transport resources, while communication delays accumulated in transmitting these signals over several hops can adversely affect the stability and robustness of any coordination scheme. Nodes are nescient if they require no *a priori* knowledge in order to execute the CCS. The extent to which this property applies determines how flexible and modular the network design can be. Decentralized content caching strategies with nescient nodes allow the network to undergo structural changes such as node additions and deletions, without requiring all nodes to be reprogrammed every time such changes occur.

A. Contributions and Related Literature

Although there is an appreciable effort within the research community to develop various networking architectures and implementation-level mechanisms that enable in-network caching, there has been little attention devoted to the development of content caching strategies with the network-level view of optimizing the use of network resources on the whole.

Existing work addressing the design of CCSs includes [3], [5], [7], [11], [15], and [18]–[23], among others.

Many papers restrict their attention to the performance of individual caches [3], [18]or network substructures such as trees or paths from sources to consumers [5], [7], [11], [19], [20]. Although the focus on tree substructures is appropriate for en-route Web-caching schemes developed for IP networking, it may lead to CCS designs that fail to leverage all of the operational features offered by ICN architectures such as content-centric networking (CCN) [6]. For example, the packetization of content in CCN allows for content to be partially requested and stored; as mentioned in [6], a destination node in a meshed content-centric network need not be restricted to assembling requested content from data packets cached by nodes that lie solely along a path toward the origin server. With appropriate interest broadcasting techniques in place, transport inefficiencies such as those described in [21] can thus be avoided.

Most approaches to the design of CCSs are based on variations of file placement policies (such as fixed probability caching), in combination with standard file eviction policies (such as least recently used or least frequently used), which are concepts borrowed from the literature on Web caching [17]. Although simple to implement, such designs are heavily based on heuristics, and the optimality of the collective behavior of caches in a general topology network is difficult to guarantee [11], [22], [23]. Sometimes these policies are tuned to optimize performance based on network models or simulations that make use of simplified traffic predictions, which may not reflect actual traffic patterns once the CCS is deployed [7], [23].

Some CCS design approaches guarantee a quantifiably suboptimal performance only in the case that certain symmetry assumptions are satisfied (such as all caches having the same size, and all content demand rates being equal at each node) [7]. Others require nodes to know or estimate the operational parameters (such as cache size) of other nodes in the network, and the efficacy with which the caching resources are utilized is known to be affected by the accuracy of such estimates [11].

In this paper, we propose a broadly applicable methodology for the design of decentralized, adaptive CCSs that systematically improve the efficiency with which a network's caching and transport resources are utilized. Although the CCSs developed here can be adapted to content delivery technologies such as CDNs and Web caching, they are designed to leverage the operational features of networks with content-aware forwarding capabilities, such as those found in CCN.

Elaborating on our work in [24], we base our CCS designs on a provably convergent, decentralized optimization algorithm called *reduced consensus optimization* (RCO), which was initially proposed in [25] (see also [26, Section 6]). In RCO, a number of nodes on a connected graph cooperate in minimizing the overall *network cost*, which encodes a set of network-wide performance objectives. This network cost is comprised as the sum of nodes' individual, privately known cost functions. There is no special structure imposed on the network topology, and the nodes are nescient with respect to the structure and size of the overall network. In particular, while most existing CCS designs consider special network structures such as trees, CCSs based on RCO allow sources, destinations, and intermediary routers to be interconnected in general mesh topologies (q.v. Fig. 1). Moreover, we assume that nodes are heterogeneous with respect to their individual caching capacities, their efficiency parameters, and the content demand rates they experience; to execute the CCS, a node does not need to know or estimate these characteristics as they pertain to other nodes in the network.

We focus on the problem of reducing the energy consumed by the network's transport and storage resources. Adopting the so-called "energy-proportional computing" model [27], we characterize the optimal network caching configuration as a minimizer of the network cost function. Each node's individual cost function depends on local content demand rates (measured in real time) and embodies the basic tradeoff induced by the energy-proportional computing model: Caching more content locally may reduce long-distance data transport costs at the expense of increased caching costs, and vice versa. In this

way, the optimal network caching configuration depends on the intensity and regional distribution of demand for content at any given time. In implementing the proposed CCS, individual nodes dynamically adjust how much and which portions of each (popular) content piece each stores, in response to their individual costs and the caching decisions made by neighboring nodes. Through nearest-neighbor interactions, the nodes collectively balance the goal of minimizing excessive copying of content throughout the network, with the goal of minimizing redundant transport of data over long distances.

Although we focus on energy efficiency, the proposed design methods are general, and can explicitly incorporate other network performance objectives as well.

Our main contributions in this paper are the following. First, we show that the decentralized content caching problem can be formulated as static *decentralized coordination control problem* (DCCP) (q.v. [26]). As such, it can be addressed by consensus optimization (CO); our second contribution is to show how CO can be used to solve the decentralized content caching problem. Third, we propose a streamlined variant of CO called *reduced consensus optimization* (RCO), which improves the CO solution by reducing the amount of communication overhead associated with coordination among nodes, and removing the need for each node to know how many other nodes are participating in the optimization. Finally, we propose a systematic method of designing nodes' individual objective functions in a way that encodes the tradeoff between caching and transport costs and induces cooperation among nodes.

An interesting paper that relates to our work is [28], where an alternative approach to optimization-based decentralized coordination control designs is presented. The work in [28] focuses on designing individual agent costs that yield a potential game with an efficient Nash equilibrium.

This paper is organized as follows. In Section II, we formulate the decentralized content caching problem as a convex program. We describe a class of consensus optimization (CO) algorithms and their convergence properties in Section III, and we consider their application to the decentralized content caching problem. In Section IV, we propose a reduced consensus optimization algorithm, which is a streamlined version of CO more suitable for the design of decentralized content caching strategies. We consider the problem of energy-efficient content delivery in Section V, and we propose a method for the design of node cost functions intended to effect desired caching behavior. In Section VI, we demonstrate in detail our proposed design methodology by means of a concrete example. We apply this methodology in Section VII to a case study involving the eleven node European optical backbone network COST239, and we evaluate the performance of the resulting content caching strategy against that of the "least frequently used" (LFU) cache policy, along several network-level performance metrics. We conclude our paper in Section VIII.

B. Notation

All vector norms $\|\cdot\|$ are Euclidean. The entry in the ith row and jth column of a matrix A is denoted $[A]_{i,j}$. For a differentiable function $J:\mathbb{R}^n \to \mathbb{R}^m$, defined by $x \mapsto J(x)$, $\nabla J:\mathbb{R}^n \to \mathbb{R}^{n\times m}$ is defined by $[\nabla J(x)]_{i,j} = \frac{\partial J_j(x)}{\partial x_i}$. We use either $\mathbf{1}$, or $\mathbf{1}_n = [1,1,\ldots,1]^T \in \mathbb{R}^n$ and either $\mathbf{0}$, or $\mathbf{0}_{m\times n}$ to denote an m by n matrix of zeros. I_n denotes the n by n identity matrix. The set of nonnegative real numbers is denoted by $\mathbb{R}_+$, and the set of positive real numbers by $\mathbb{R}_{++}$. If S is a set, then $|S|$ is its cardinality.

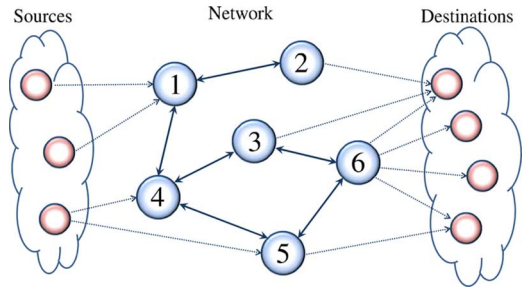


Fig. 1. Each node in the network can access files from the source repositories. In general, the source nodes, the destination nodes, and the intermediary routers can be interconnected over a general mesh topology. The ith node measures a demand $d_{i,k}$ for content piece k.

II. Description of the Adaptive Content Caching Problem

We consider an abstract network of N nodes that may exchange information over a given connected, undirected graph $\mathcal{G}_C = (\mathcal{V}, \mathcal{E}_C)$, where $\mathcal{V} = \{1,\ldots,N\}$ is the set indexing the nodes and $\mathcal{E}_C \subset \mathcal{V}\times\mathcal{V}$ is the set of links between them.

We assume that there is a set of content repository nodes (or sources) that generate and store permanent copies of a large number of files that may need to be accessed throughout the network, as shown in Fig. 1. This set of source nodes may include some of the nodes within $\mathcal{G}_C$, or it may be entirely external to $\mathcal{G}_C$. However, all nodes within $\mathcal{G}_C$ are able to access any file stored at any source node.

We let $\mathcal{F} = \{1,\ldots,F\}$ denote the set of F most popular content pieces that are to be cached throughout the network. Each content piece may represent a single file, or an aggregate of related files with similar popularities. At each instant, the ith node in the network experiences a demand for content piece $k \in \mathcal{F}$ that can be calculated as a windowed average request rate—i.e., supposing that $(t_{n_k^i})_{n_k^i=1}^{\infty} \subset \mathbb{R}_+$ denotes the sequence of time instants at which requests for content piece k arrive at node i, the demand for k at node i can be calculated as

$$d_{i,k}(t) = \frac{1}{W}\int_{t-W}^{t} \sum_{n_k^i=1}^{\infty} \delta\left(t - t_{n_k^i}\right) dt \tag{1}$$

where $\delta(\cdot)$ is the Dirac delta function, and W is the window size in units of time. Intuitively, for a given traffic pattern observed at node i, selecting larger values for W yields a set of signals $d_{i,k}(t)$ exhibiting less volatility.

The signal $d_{i,k}$, which can be measured by node i, represents an aggregated request rate that originates either from a pool of users directly connected to node i, or from other nodes within the network (q.v. Fig. 1).

We assume that each content piece is divided into P_k packets of q data units in size and that each node $i \in \mathcal{V}$ is equipped with a cache of size $B_i q$ units of data.

Moreover, these packets can be individually requested from the source repository or other nodes. This mechanism allows each node to store selected portions of various content pieces, if desired.

Given a communication graph $\mathcal{G}_C = (\mathcal{V}, \mathcal{E}_C)$, the content catalog $\mathcal{F}$, content sizes P_k, $\forall k \in \mathcal{F}$, cache sizes B_i, $\forall i \in \mathcal{V}$, and the set of demand rates $d_{i,k}(t)$, the adaptive

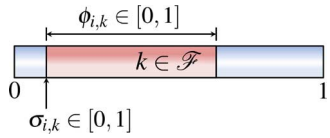


Fig. 2. Router i stores a contiguous block of content piece k, starting at $\sigma_{i,k}$, and having a size of $\phi_{i,k} P_k q$ units of data.

content-caching problem is to decide which portions of which content pieces each node ought to cache at time t. One straightforward way to encode such a decision is to assume that nodes store only contiguous blocks of a content piece. In that case, the decision can be characterized in terms of only two numbers: $\sigma_{i,k} \in \{\frac{1}{P_k}, \ldots, \frac{P_k-1}{P_k}, 1\}$, indicating the location at which node i *starts* to store its contiguous block of $k \in \mathcal{F}$, and $\phi_{i,k} \in \{\frac{1}{P_k}, \ldots, \frac{P_k-1}{P_k}, 1\}$, the *fraction* of the whole content piece that this block represents. Since $\forall k \in \mathcal{F}, P_k$ is likely a large integer, we may approximate $\sigma_{i,k}$ and $\phi_{i,k}$ by allowing them to take values in the real unit interval, as shown in Fig. 2.

Collecting these variables, we let

$$x_i = [\sigma_{i,1}, \phi_{i,1}, \ldots, \sigma_{i,F}, \phi_{i,F}]^T \in \mathbb{R}^{2F} \qquad \forall i \in \mathcal{V} \tag{2}$$

denote the vector of node i's caching decisions, and we refer to $x = [x_1^T, \ldots, x_N^T]^T \in \mathbb{R}^{2NF}$ as the *network caching configuration*.

With this framework, the adaptive content caching problem can be rephrased as follows: What constitutes the best network caching configuration? What is deemed "best" depends on the set of network performance objectives, which may include factors such as minimization of network congestion and improvement of load balancing, reduction of access latency, enforcement of various QoS measures for different classes of content, robustness to transport disruptions and node failures, elimination of redundant traffic flows, minimization of energy costs, and others. Such performance objectives can often be expressed within the formalism of convex optimization. We therefore assume that there exists a network caching configuration $x^* \in \mathbb{R}^{2NF}$ that best meets a given set of network performance criteria, and that x^* can be expressed as a solution to an optimization problem of the form

$$\min_{x \in \mathbb{R}^{2NF}} \quad J(x) \tag{3}$$

$$\text{s.t.} \quad \sigma_{i,k} \geq 0 \qquad \forall i \in \mathcal{V}, \forall k \in \mathcal{F} \tag{4}$$

$$\phi_{i,k} \geq 0 \qquad \forall i \in \mathcal{V}, \forall k \in \mathcal{F} \tag{5}$$

$$\sigma_{i,k} + \phi_{i,k} \leq 1 \qquad \forall i \in \mathcal{V}, \forall k \in \mathcal{F} \tag{6}$$

$$\sum_{k=1}^{F} \phi_{i,k} P_k \leq B_i \qquad \forall i \in \mathcal{V} \tag{7}$$

where (7) represents node i's cache capacity constraint, and inequalities (4)–(6) are required by the manner in which the decision variables are defined (q.v. Fig. 2). Furthermore, we assume that the function $J : \mathbb{R}^{2NF} \to \mathbb{R}$ can be written as

$$J(x) = J_1(x) + \cdots + J_N(x) \tag{8}$$

where $J_i(x)$ represents a measure of node i's individual performance, which may in general depend on the entire network caching configuration x.

Our aim now is to develop a set of rules by which the nodes may update their individual caching decisions toward collectively achieving the optimal network configuration x^*. These update rules are required to be decentralized and nescient, meaning that nodes are required to update their decisions through nearest-neighbor interactions, and using locally available information only. Information that is considered locally available to node i includes knowledge of the analytic structure of its individual cost function $J_i(\cdot)$, the demand rates $d_{i,k}(t)$ (and file sizes P_k), $\forall k \in \mathcal{F}$, and knowledge of its neighbors' caching decisions. In particular, nodes should not require knowledge of the entire network configuration $x(t)$ at any time, since such knowledge generally necessitates multihop communication, adding to the coordination overhead and potentially incurring detrimental delays in the coordination signals.

The decentralized content caching problem formulated in this section is an instance of a class of *static, decentralized coordination control problems* (static DCCPs) defined in [26, Sec. 1.2]. Formulating the decentralized content caching problem as a static DCCP is the first of our four main contributions in this paper.

Since the decentralized content caching problem is a static DCCP, the required update rules can be developed on the basis of a class of decentralized optimization algorithms known as *consensus optimization* (CO). In the sequel, we describe the operation and convergence properties of such schemes. Our second main contribution is to show how such schemes can be used to solve the decentralized content caching problem. We then propose a streamlined version of CO called *reduced consensus optimization*, which improves on the solution provided by CO. We return to the problem of designing CCSs in Section V, where we consider the minimization of the network's energy consumption as the primary performance objective.

III. Consensus Optimization

CO methods originate in the work of Tsitsiklis [29]and are further developed by Nedić and coworkers in [30]. The problem setting considered in [30] involves N abstract agents who are able to communicate with one another over an undirected graph $\mathcal{G}_C = (\mathcal{V}, \mathcal{E}_C)$, and who are to collaboratively solve a convex optimization problem of the form

$$\min_{x \in \chi} \quad \sum_{i=1}^{N} J_i(x), \quad \text{where} \quad \chi = \cap_{i=1}^{N} \chi_i \tag{9}$$

and agent i has knowledge only of the set $\chi_i \subset \mathbb{R}^n$ and the function $J_i : \mathbb{R}^n \to \mathbb{R}$. In this setting, agent i updates its estimate $\xi_i \in \mathbb{R}^n$ of some optimizer $x^* \in \mathbb{R}^n$ that solves problem (9), according to the algorithm

$$\xi_i(t+1) = \mathbf{P}_{\chi} \left[\sum_{j=1}^{N} [A]_{i,j} \xi_j(t) - \alpha s_i(t) \right], \qquad t \in \mathbb{N} \tag{10}$$

where $\mathbf{P}_{X_i}(\cdot)$ is the orthogonal projection operator defined as

$$\mathbf{P}_{\chi_i}(p) = \arg\min_{w \in \chi_i} \|p - w\| \tag{11}$$

$s_i(t)$ is any subgradient of $J_i(\cdot)$ at $\xi_i(t)$, and $A \in \mathbb{R}^{N \times N}$ is a specially weighted adjacency matrix associated to $\mathcal{G}_C$. The convergence rate properties of (10) are studied in [30], and its stability properties are established in [26, Sec. 3].

We briefly summarize a set of sufficient conditions under which the algorithm (10) is shown in [26] to converge to an arbitrarily small neighborhood of the set X^* of optimal solutions to (9).

1) Assumptions on the Communication Structure: The most important technical assumption concerns the weighting of the edges of the graph $\mathcal{G}_C$. The weighted adjacency matrix $A \in \mathbb{R}^{N\times N}$, whose elements are the coefficients with which each agent combines its estimate of x^* with its neighbors' estimates in (10), is assumed to have the following properties.

A3.1: The matrix A is stochastic, symmetric and primitive.[1] $\diamondsuit$

The intuitive meaning and the technical implications of A3.1 are discussed in [26, Sec. 2.2], for example.

There are several methods by which the edge weights can be assigned so that the adjacency matrix A satisfies A3.1 [31]. One simple decentralized process by which all the nodes in $\mathcal{G}_C$ can select a set of edge weights satisfying A3.1 in only four steps is given by Algorithm 3.1 below. Letting

$$\mathcal{N}_C(i) = \{j \in \mathcal{V}\backslash\{i\} | (j,i) \in \mathcal{E}_C\} \tag{12}$$

denote the set of agent i's neighbors on $\mathcal{G}_C$, this algorithm proceeds as follows.

Algorithm 3.1:

1) $\forall i \in \mathcal{V}$, agent i sets $p_i = \frac{1}{|\mathcal{N}_C(i)|+1}$.
2) $\forall i \in \mathcal{V}$, agent i sends to each agent $j \in \mathcal{N}_C(i)$ the number p_i, and receives the number p_j in return.
3) $\forall (i,j) \in \mathcal{E}_C$, agents i and j select the weight for edge (i,j) as $[A]_{i,j} = [A]_{j,i} = \min\{p_i, p_j\}$.
4) $\forall i \in \mathcal{V}$, agent i sets $[A]_{i,i} = 1 - \sum_{j\in\mathcal{V}\backslash\{i\}} [A]_{i,j}$

A similar process is described in [32, Sec. III.B].

2) Assumptions on the Cost Structure: Just as for the steepest descent algorithm, convergence proofs for (10) rely on certain properties of the costs $J_i(\,\cdot\,)$. For the purpose of designing decentralized content caching strategies, we can assume that the costs $J_i(\,\cdot\,)$ are differentiable and that in (10)

$$s_i(t) = \nabla J_i(\xi_i(t)). \tag{13}$$

In that case, the conditions considered [26](q.v. Section 3.3.2) may be specialized to the following:

A3.2: For each $i \in \mathcal{V}$, the gradient of $J_i(\,\cdot\,)$ is locally Lipschitz continuous—i.e., there exists a real number $L_i > 0$ such that for any p and w in $\mathbb{R}^n$

$$\|\nabla J_i(p) - \nabla J_i(w)\| \leq L_i \|p - w\|. \tag{14}$$

Moreover, the network cost $J(\,\cdot\,)$ us convex. $\diamondsuit$

Under conditions A3.1 and A3.2, it is shown in [26, Sec. 3] that the set $\prod_{i=1}^N X^*$ [where X^* is the set of solutions to problem (9)] is *semiglobally, practically asymptotically stable* for the algorithm (10). In other words, for any arbitrarily large set of initial conditions $\xi_i(0)$ and for any arbitrarily small neighborhood $B_\rho^n(X^*)$ of X^*, there exists a number $\bar{\alpha} \in \mathbb{R}_{++}$ such that whenever the step size $\alpha \in (0, \bar{\alpha})$, each sequence $(\xi_i(t))_{t=0}^\infty$ asymptotically approaches $B_\rho^n(X^*)$. Moreover, for any $\alpha \in (0, \bar{\alpha})$, the ultimate upper bound on the agents' estimation errors is given by

$$\limsup_{t\to\infty} \|\xi(t) - \mathbf{P}_{\mathbf{X}^*}(\xi(t))\|^2 \leq \alpha^2 K_o \tag{15}$$

where $\mathbf{X}^* = \prod_{i=1}^N X^*$, $\xi(t) = [\xi_1(t)^T, \ldots, \xi_N(t)^T]^T \in \mathbb{R}^{Nn}$, and K_o is a constant whose value depends on the problem parameters N, α, L_i and the second largest eigenvalue $\lambda_2(A)$ of the matrix A.

[1] A nonnegative matrix is *stochastic* if each of its rows sums to unity. A matrix is *primitive* if it is irreducible and it has a positive, real eigenvalue $\lambda_1(A)$ equal to its spectral radius, and this eigenvalue has a modulus strictly larger than that of any other eigenvalue of A. If A is a weighted adjacency matrix corresponding to a graph $\mathcal{G}_C = (\mathcal{V}, \mathcal{E}_C)$, then A is *irreducible* if and only if $\mathcal{G}_C$ is connected. Please see [26, Sec. 2.2] for more details.

A. Application of CO to Decentralized Content Caching

One way to use algorithm (10) to solve content caching problems of the form (3) to (7) is to interpret $\xi_i(t) \in \mathbb{R}^{2NF}$ as node i's estimate at time t of some optimal network caching configuration $x^* \in \mathbb{R}^{2NF}$. With this interpretation, one may take $x_i(t) = \xi_{i,i}(t) \in \mathbb{R}^{2F}$ to represent node i's implemented caching decision at time t, and $\xi_{i,j} \in \mathbb{R}^{2F}$ to represent node i's "suggestion" to node j on how to behave. The constraint sets χ_i in (10) could then be related to the constraints (4) to (7) by taking

$$\chi_i = \tilde{X}_{i,1} \times \cdots \times \tilde{X}_{i,N} \tag{16}$$

where

$$\tilde{X}_{i,j} = \begin{cases} X_i, & j = i \\ X_0, & \text{otherwise} \end{cases} \tag{17}$$

and X_0 and X_i are given by

$$X_0 = \left\{ x_i \in \mathbb{R}^{2F} \,|\sigma_{i,k} \geq 0,\ \phi_{i,k} \geq 0,\ \sigma_{i,k} + \phi_{i,k} \leq 1] \quad \forall k \in \mathcal{F} \right\}$$
$$X_i = X_0 \cap \left\{ x_i \in \mathbb{R}^{2F} \,\middle|\, \sum_{k\in\mathcal{F}} \phi_{i,k} P_k \leq B_i \right\} \quad \forall i \in \mathcal{V}. \tag{18}$$

In this way, node i enforces only its own cache capacity constraint, and need not be aware of other nodes' cache sizes.

Another favorable aspect of our interpretation is that in order to implement algorithm (10), a node does not require knowledge of other nodes' caching decisions, except those in its graphical neighborhood. However, each node needs to know how many other nodes are participating, since it must maintain an estimate of each component of the optimal network configuration x^*. This is a cumbersome requirement; each time the network undergoes a structural modification such as the addition of a node, each preexisting node must modify its decision updating rule to include additional real variables representing an estimate of the added node's optimal caching decision. Moreover, the dimension of each vector ξ_i updated via (10) grows linearly in N, and therefore so does the communication overhead associated with coordinating the node's caching decisions.

Motivated by these considerations, we develop a streamlined version of CO in which node i need not necessarily maintain an estimate of every other node's optimal caching decision. We refer to this algorithm as *reduced consensus optimization* (RCO) (q.v. [26, Section 6]).

IV. Reduced Consensus Optimization

With the interpretation of the variables ξ_i given in Section III-A, it seems excessive to have node i maintain an estimate of node j's optimal caching decision for all $j \in \mathcal{V}\backslash\{i\}$, especially if the caching decisions of node j have a negligible effect on node i's private performance measure $J_i(\,\cdot\,)$. In large

networks, the decisions of distant nodes may well have little or no impact on local performance measures.

For this reason, we begin by introducing the *interference digraph* $\mathcal{G}_I = (\mathcal{V}, \mathcal{E}_I)$ whose edge set we define as

$$\mathcal{E}_I = \{(j,i) \in \mathcal{V} \times \mathcal{V} | \nabla_{x_j} J_i(x) \not\equiv 0\}. \tag{19}$$

In other words, (j,i) is an edge in the interference graph if the decisions of node j influence the cost experienced by node i. We do not assume any particular relationship between the communication graph $\mathcal{G}_C$, and the interference graph $\mathcal{G}_I$.

Next, for each $j \in \mathcal{V}$, we define $I(j)$ as the set of all nodes $i \in \mathcal{V}$ whose costs are affected by the decisions of node j—i.e.,

$$I(j) = \{i \in \mathcal{V} | (j,i) \in \mathcal{E}_I\} \cup \{j\}. \tag{20}$$

For each $j \in \mathcal{V}$, we then form the graph $\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j)$, which is any smallest connected subgraph of $\mathcal{G}_C$ containing all the nodes in $I(j)$. Constructing $\mathcal{G}_j$ is always possible since $\mathcal{G}_C$ is assumed to be connected (q.v. A3.1). In general therefore, $\mathcal{E}_j \subseteq \mathcal{E}_C$ and $I(j) \subseteq \mathcal{V}_j \subseteq \mathcal{V}$. However, the choice of $\mathcal{G}_j$ may not be unique, since the shortest path connecting any $k, i \in I(j)$ may not be unique.

We let

$$\mathcal{N}_j(i) = \{k \in \mathcal{V}_j \backslash \{i\} \,|\, (k,i) \in \mathcal{E}_j\}$$

denote the set of agent i's neighbors on $\mathcal{G}_j$. Finally, with the set

$$S(i) = \{j \in \mathcal{V} | i \in \mathcal{V}_j\} \tag{21}$$

we identify those subgraphs $\mathcal{G}_j$ to which agent i belongs.

Then, for all $t \in \mathbb{N}$, the (unconstrained) RCO algorithm is implemented by each agent $i \in \mathcal{V}$ as

$$\begin{aligned} &\xi_{i,j}(t+1) \\ &\quad = \begin{cases} \mathbf{P}_{\tilde{X}_{i,j}}\Big[\sum_{k=1}^N a_{j,k}^{(i)} \xi_{k,j}(t) - \alpha \nabla_{x_j} J_i(\xi_i(t))\Big], & j \in S(i) \\ 0, & j \in \mathcal{V} \backslash S(i) \end{cases} \\ &x_i(t) = \xi_i(t) \end{aligned} \tag{22}$$

where $\tilde{X}_{i,j}$ relates to χ_i as in (16), and

$$a_{j,k}^{(i)} = \begin{cases} \min\left\{\frac{1}{|\mathcal{N}_j(i)|+1}, \frac{1}{|\mathcal{N}_j(k)|+1}\right\}, & \text{if } k \in \mathcal{N}_j(i) \\ 1 - \sum_{m=1}^N a_{j,m}^{(i)}, & \text{if } k = i \text{ and } i \in \mathcal{V}_j \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

Fig. 3 illustrates the notion of the interference structure, the construction of the subgraphs $\mathcal{G}_j$ and the definition of the sets $S(i)$ for a simple four-node network.

The intuition behind RCO is that each subgraph $\mathcal{G}_j$ has a consensus matrix $A_j \in \mathbb{R}^{|\mathcal{V}_j| \times |\mathcal{V}_j|}$ associated to it, with individual entries assigned according to Algorithm 3.1. The set $\mathcal{V}_j$ identifies those nodes that update an estimate of j's optimal decision x_j^*.

Remark 4.1: The main difference between RCO and the standard CO algorithm (10) is that node i need not maintain estimates of *all* other nodes' optimal decisions. In (22), node i updates an estimate of precisely those components of x^* associated to all nodes $j \in S(i)$. This feature is particularly advantageous when the interference structure is sparse since the set $S(i)$ could have a much smaller cardinality than the set $\mathcal{V}$. Consequently, for the case in which $|S(i)| \ll N$, RCO requires significantly fewer real-number updates and exchanges among the nodes at each iteration, thereby reducing the communication and processing overhead associated with coordinating the nodes' caching decisions. In Section VI-A, we quantify this reduction for an RCO-based CCS strategy designed to improve energy efficiency in the four-node example network shown in Fig. 3.

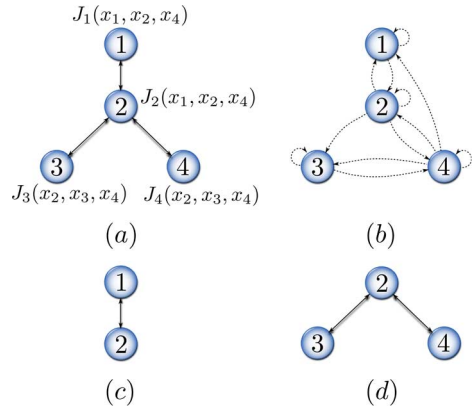


Fig. 3. (a) Network with $\mathcal{G}_C$ and agents' individual costs and (b) the associated interference structure. The subgraphs $\mathcal{G}_2 = \mathcal{G}_4 = \mathcal{G}_C$, whereas $\mathcal{G}_1$ and $\mathcal{G}_3$ are shown in (c) and (d), respectively. Each subgraph $\mathcal{G}_j$ is constructed by including all the nodes and edges along a shortest path from node j to each node i, whose cost is affected by the actions of node j.

Another important advantage of RCO over (10) is that node i need not be aware of any node $j \in \mathcal{V} \backslash S(i)$. In other words, whenever $S(i) \neq \mathcal{V}$, agent i is *nescient* with respect to the number of agents on the network N, and the network's overall interconnection structure $\mathcal{G}_C$. As a consequence, agent i's update rule does not change when nodes are added or taken out of the network, so long as the set $S(i)$ is unaffected by the change. ♢

Remark 4.2: The convergence rate analysis of both CO and RCO can be carried out by means of the Lyapunov techniques proposed in the proof of [26, Theorem 2.5.1]. The analytical arguments outlined in [25] lead to the same structural form for the difference equation governing the evolution of the Lyapunov function used to study the convergence properties of CO, as for that used to study the convergence properties of RCO. The upper bound on the convergence rates that can be derived for RCO is therefore expected to be of the same order as that for CO. ♢

V. Content Caching Strategies For Energy-Efficient Content Delivery

We return now to the adaptive content caching problem (3)–(7). Building on the work in [12]–[15], we address the problem energy-efficient content distribution over networks with in-network caching capabilities.

As in [12]–[15], we adopt the *energy-proportional computing model* [27], in which it is assumed that the energy consumption of network transport and routing equipment is proportional to its utilization. In [12]–[15], the transport resources considered include transmission, routing, and switching equipment.

In adopting this model, we approximate the amount of energy consumed by a node as being proportional to the amount of data it caches, in addition to the amount of data it transports from

other nodes toward the consumer. Depending on whether a node is a core, edge, or access router, its energy efficiency profile is expected to be different; we account for this heterogeneity by allowing the costs to depend on $E_{\mathrm{ca},i}$, the amount of power (in watts) that node i consumes by caching q bits of data, and $E_{\mathrm{tr},i}$, the amount of energy (in joules) that node i consumes by transporting q bits of data.

Though the energy-proportional computing model is not without criticism (since devices tend to consume some amount of energy even when idling), studies such as those in [33] and [14] suggest that its adoption suffices for our purposes.

In the context of content distribution, the energy-proportional model induces a basic performance tradeoff: While transport energy is reduced by caching as much content as possible close to consumer demand, caching more content consumes more caching energy. To capture this tradeoff, we propose that individual agents' caching behavior should be governed by the following set of operational principles:

Operational Principles

1) The fraction $\phi_{i,k}(t)$ of content piece k that node i caches should be positively correlated with $d_{i,k}(t)$—i.e., node i should cache more of the content that is in high demand, and less of the content that is not in demand.
2) To avoid excessive copying of content within the network, node i should avoid caching the same portions of the same content that nearby nodes cache.
3) When $d_{i,k}(t)$ is high, some nodes in i's graphical vicinity should coordinate among themselves to maximize their collective caching coverage of content piece k, so as to make those portions not cached by i available for short-distance transport to i.

These operational principles are to be encoded into the update rules governing individual agents' caching decisions by means of appropriately designed individual cost functions $J_i(\cdot)$. There are many degrees of freedom associated with designing a cost and interference structure in order to enforce these principles. We introduce the notions of *segmentation* and *clustering* and propose their use in guiding the design process.

A. Segmentation and Clustering

Let each content piece be divided into M *segments*, where $M \leq N$ is a design parameter. Then, to each node $i \in \mathcal{V}$ assign a number $s_i \in \{1, \ldots, M\}$, meaning that node i caches a portion of each content piece corresponding to the s_ith segment. Next, to each node i, assign a *cluster* of $M-1$ nodes $C_i \subset \mathcal{V}\backslash\{i\}$ such that for all distinct k and j in $C_i \cup \{i\}$, $s_k \neq s_j$. In other words, each node in each cluster is assigned to a different segment of content.

Let $H_{i,j}$ be the number of hops separating nodes i and j along a shortest path between them over $\mathcal{G}_C$, and let $H_i = \max_{j \in C_i} H_{i,j}$. We identify the number of segments M and the assignment of clusters $C_i, \forall i \in \mathcal{V}$, as important degrees of freedom in the design process. Although in this paper we do not address the question of how to optimally assign segments and clusters to each node, we suggest that for a given network topology $\mathcal{G}_C$, these should be chosen so as to minimize H_i, for each i. However, we note that for a fixed M, the problem of minimizing H_i via segmentation and cluster assignments for a given network topology can be framed as a graph coloring problem.

B. Designing the Costs $J_i(\cdot)$

With M and C_i specified, we consider the following general structure for the cost functions:

$$J_i(x) = \sum\nolimits_{k \in \mathcal{F}} \Bigg[E_{\mathrm{ca},i}\phi_{i,k}P_k + E_{\mathrm{tr},i}d_{i,k}(t)(1-\phi_{i,k})P_k + d_{i,k}(t)c_i \left(\left(\sum\nolimits_{j=1}^{M} \phi_{\pi_i(j),k} \right) - 1 \right)^2 + b_{i,1}\sigma^2_{\pi_i(1),k} + \sum_{j=1}^{M-1} b_{i,j+1}(\sigma_{\pi_i(j),k} + \phi_{\pi_i(j),k} - \sigma_{\pi_i(j+1),k})^2 \Bigg] \tag{24}$$

where $\pi_i : \{1, \ldots, M\} \to C_i \cup \{i\}$ is the permutation defined as $\pi_i : s_k \mapsto k \in \mathcal{V}$, and $b_{i,j}, j \in \{1, \ldots, M\}$ and c_i are positive real tuning parameters. Node i is to minimize its individual cost $J_i(\cdot)$ by attempting to influence the behavior of all other nodes that affect it. The rationale for the proposed cost design can be explained as follows. The first term in (24) penalizes node i for expending caching energy in proportion to the amount of data cached, while the second term is a penalty for having to transport that content which is not being cached. The second term encourages node i to cache larger portions of that content which is in high demand, in accordance with the first operational principle. On the other hand, when $d_{i,k} = 0$, the only term pertaining to content k in $J_i(x)$ is the caching penalty term, prompting node i to cache less of the content that is not in demand. The first two terms thereby capture the basic performance tradeoff induced by the energy-proportional computing model. The remaining terms promote cooperation among nodes, as per the second and third operational principles.

The fourth and fifth terms in (24) are intended to encourage nodes to maintain caching boundaries between their respective content segments. We therefore refer to these as the *segmentation boundary* terms. Ideally, the contiguous block of content piece k that node $\pi_i(j+1) \in C_i \cup \{i\}$ caches should begin precisely where the block cached by node $\pi_i(j)$ ends—for all segments $j \in \{1, \ldots, M-1\}$. When this is the case, the fifth term in (24) is identically zero. The fourth term encourages those nodes to whom the first segments are assigned to cache all content pieces starting with the first packet. Together, these terms have two functions. First, they penalize excessive copying of content within a cluster. Second, they penalize caching "gaps," thereby promoting the caching coverage of content within a cluster in a systematic way.

The maximization of content coverage is promoted further by the third term in (24), to which we refer as the *coverage* term. This term allows node i to encourage other nodes within the cluster C_i to maximize the collective caching coverage of each content piece k, in proportion to its demand. The fact that this term is proportional to $d_{i,k}(t)$ reflects the importance that node i attributes to this task; when content k is in high demand, it is very important to node i that nearby nodes cooperate in caching k in its entirety, if possible. This term is identically zero when each content piece is cached in entirety within the cluster C_i, thereby enforcing the third operational principle.

We note that this cost structure does not preclude either the extreme possibility that a given content piece k is not cached at all within the network, or that each node in the network caches

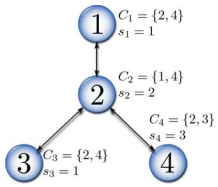


Fig. 4. Segmentation assignment for the graph $\mathcal{G}_C$ in the design example.

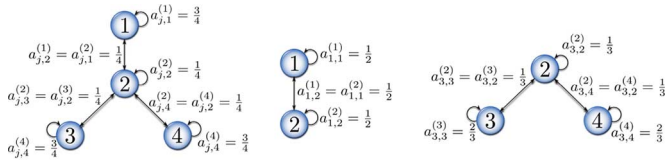


Fig. 5. Assigning the weights $a_{j,k}^{(i)}$ to each subgraph $\mathcal{G}_j$, according to (23). Edge weights for subgraphs $\mathcal{G}_2$ and $\mathcal{G}_4$ are shown in (a), those for subgraph $\mathcal{G}_1$ in (b), and those for subgraph $\mathcal{G}_3$ in (c). These same edge weights can be derived by applying Algorithm 3.1 to each subgraph individually.

a complete copy of content k (provided that $P_k \leq B_i, \forall i \in \mathcal{V}$). If demands $d_{i,k}(t)$ fall to zero for all i, then we expect that eventually $(\sigma_{i,k}, \phi_{i,k}) \to (0,0)$ as well, due to the first and fifth terms in (24). The qualitative characteristics of collective network behavior resulting from the proposed costs is explored further in Section VI-A.

VI. Design Example

Combining the ideas introduced in Sections IV and V, we now show in detail how to apply our proposed design methodology. For the sake of concreteness, let us consider the example network shown in Fig. 3(a), and a content catalog $\mathcal{F} = \{1,2\}$ containing only two content pieces. Node i's caching decision at time t is then given by

$$x_i(t) = [\sigma_{i,1}(t), \phi_{i,1}(t), \sigma_{i,2}(t), \phi_{i,2}(t)]^T, \qquad i \in \{1,2,3,4\}. \tag{25}$$

We begin by specifying the sets $S(i)$ and the weights $a_{j,k}^{(i)}$ that are needed for implementing (22). We choose $M = 3$ and assign the segments and clusters as indicated in Fig. 4. With these segment and cluster assignments, the costs (24) ascribed to each node in $\mathcal{G}_C$ induce the interference structure shown in Fig. 3(b)—namely

$$I(1) = \{1,2\} \quad I(2) = \{1,2,3,4\}$$
$$I(3) = \{3,4\} \quad I(4) = \{1,2,3,4\}.$$

Fig. 3(c) and (d) respectively shows the smallest subgraphs $\mathcal{G}_1$ and $\mathcal{G}_3$ containing the sets $I(1)$ and $I(3)$. Therefore, we have

$$S(1) = \{1,2,4\} \quad S(2) = \{1,2,3,4\}$$
$$S(3) = \{2,3,4\} \quad S(4) = \{2,3,4\}.$$

In this example, $S(i) = C_i \cup \{i\}$ for all i except $i = 2$. Recalling that the set $S(i)$ identifies those components of x^* that agent i estimates through the RCO update rule (22), we note that node 2 must maintain an estimate of x_3^*—not because node 3 affects its cost, but because node 2 must act as an information conduit between nodes 3 and 4, who interfere with one another.

The values of the weights $a_{j,k}^{(i)}$ may be chosen as shown in Fig. 5.

We let

$$\xi_{i,j}(t) = \left[\sigma_{j,1}^{(i)}(t), \phi_{j,1}^{(i)}(t), \sigma_{j,2}^{(i)}(t), \phi_{j,2}^{(i)}(t)\right]^T \in \mathbb{R}^4 \tag{26}$$

denote node i's estimate at time t of $x_j^* \in \mathbb{R}^4$, node j's optimal caching action. The RCO-based content-caching strategy for this example then takes the form

$$\xi_{i,j}^+ = \begin{cases} \mathbf{P}_{X_0}\left[\sum_{k=1}^N a_{j,k}^{(i)} \xi_{k,j} - \alpha \nabla_{x_j} J_i(\xi_i)\right], & j \in S(i)\backslash\{i\} \\ \mathbf{P}_{X_i}\left[\sum_{k=1}^N a_{j,k}^{(i)} \xi_{k,j} - \alpha \nabla_{x_j} J_i(\xi_i)\right], & j = i \\ 0, & j \in \mathcal{V}\backslash S(i) \end{cases}$$
$$x_i = \xi_{i,i} \tag{27}$$

where the weights $a_{j,k}^{(i)}$ are as shown in Fig. 5, the sets X_0 and X_i are as in (18), the cost functions $J_i(\cdot)$ are as in (24), and we have used the shorthand $\xi_{i,j}^+$ to denote $\xi_{i,j}(t+1)$.

There are several numerical methods available for the computation of the projection (11). Since the sets X_0 and $X_i, \forall i \in \mathcal{V}$, defined in (18) are polytopes in $\mathbb{R}^4$, each defined as the intersection of finitely many half-spaces, the projection operation may be implemented using the algorithm described in [34], for example. That algorithm is implemented as follows. Let X be a polytope in $\mathbb{R}^m$, described as the intersection of h half-spaces—i.e.,

$$X = \{x \in \mathbb{R}^m \,|\, Hx \preceq b\} \tag{28}$$

where $H \in \mathbb{R}^{h \times m}$, $b \in \mathbb{R}^h$, and "$\preceq$" denotes a component-wise inequality.

We denote by $(X)_j$ the jth half-space

$$(X)_j = \{x \in \mathbb{R}^m \,|\, [H]_j x \leq b_j\}, \qquad j \in \{1, \ldots, h\} \tag{29}$$

so that $X = \bigcap_{j=1}^m (X)_j$. Then, given some point $z \in \mathbb{R}^m$, its projection onto X can be iteratively computed as follows:

Algorithm 6.1 ([34, Algorithm 1]):

1) Let $p_1(0), \ldots, p_h(0)$ be arbitrary points in $\mathbb{R}^m$.
2) Let $x(0) = z - \sum_{j=1}^h p_j(0)$.
3) Let $q_j(0) = x(0) + \lambda p_j(0)$.
4) For $k \in \mathbb{N}$, let:
   a) $p_j(k) = \frac{1}{\lambda}[q_j(k-1) - \mathbf{P}_{(X)_j}(q_j(k-1))], \forall j \in \{1, \ldots, h\}$.
   b) $x(k) = z - \sum_{j=1}^h p_j(k)$.
   c) $q_j(k) = x(k) + \lambda p_j(k)$.

where $\lambda > 0$ is a tuning parameter. According to [34, Theorem 1]

$$\lim_{k \to \infty} x(k) = \mathbf{P}_X(z) \tag{30}$$

whenever λ is chosen to be larger than $h/2$.

In Step 4a of this algorithm, we are required to compute the projection of each point $q_j(k-1), j \in \{1, \ldots, h\}$ onto the half-space $(X)_j$; fortunately, $\mathbf{P}_{(X)_j}(\cdot)$ can easily be derived in closed form (q.v. [35, Sec. 8.1.1], for example). A more detailed discussion of the implementation of Algorithm 6.1 for this example is given in [26, Sec. 7].

A. Qualitative Behavior of the RCO-Based CCS

We simulate algorithm (27) with the costs $J_i(x)$ defined as in (24), for the network shown in Fig. 4, and its segments and clusters assigned as indicated in Fig. 4. We chose the following parameters to obtain the simulation results shown in this section:

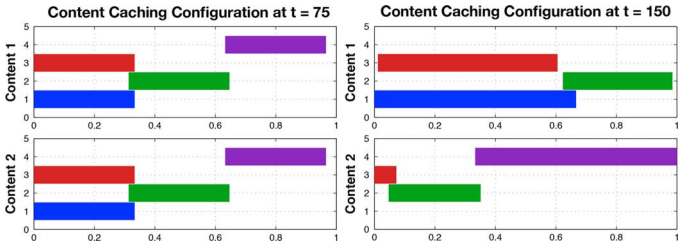


Fig. 6. Content caching configurations at $t = 75$ and $t = 150$. The top and bottom plots indicate the starting location $\sigma_i^{(i)}(t)$ and the fraction $\phi_i^{(i)}(t)$ of the contiguous block of content that each node $i \in \{1, 2, 3, 4\}$ caches, for content pieces 1 and 2, respectively. The nodes are labeled along the vertical axis.

- Router cache sizes: $B_1 = B_2 = B_3 = B_4 = 1000$ (i.e., router i has a cache size of $B_i q$ data units, where q is the number of data units comprising one data packet).
- Content catalog: $\mathcal{F} = \{1, 2\}$, with $P_1 = 1500$ and $P_2 = 1500$.
- Segmentation boundary terms: $b_{i,j} = 200$, for all $i \in \{1, 2, 3, 4\}$ and $j \in \{1, 2, 3\}$.
- Coverage terms: $c_1 = c_2 = c_3 = c_4 = 50$.
- Transport efficiencies: $E_{\mathrm{tr},1} = E_{\mathrm{tr},2} = E_{\mathrm{tr},3} = E_{\mathrm{tr},4} = 1.5$ J per q units of data.
- Caching efficiencies: $E_{\mathrm{ca},1} = E_{\mathrm{ca},2} = E_{\mathrm{ca},3} = E_{\mathrm{ca},4} = 50$ W per q units of data.
- Step size: $\alpha = 4.8 \times 10^{-6}$.
- Initial conditions: $\xi_{i,j}(0) = [0, 0, 0, 0]^T$, for all $i \in \{1, 2, 3, 4\}$ and $j \in S(i)$.

To implement Algorithm 6.1, we took $\lambda = 5$ and generated h random vectors in $p_j(0) \in [0, 1]^4$, $j \in \{1, \ldots, h\}$, where $h = 6$ for X_0 defined in (18) and $h = 7$ for the sets X_1 to X_4.

To investigate the adaptive capability of the proposed decentralized caching strategy, we changed the content demand rates $d_{i,k}(t)$ at time $t = 75$. Initially all the demands are identical for both content pieces and all nodes. At $t = 75$, node 1 experiences an increase in demand for content piece 1 and a decrease in demand for content piece 2, while node 4 experiences the exact opposite change in demand, and demands remain the same for other nodes.

The results are shown in Figs. 6 and 7. Fig. 6 shows the network caching configuration at $t = 75$ and $t = 150$, while Fig. 7 shows the time evolution of the network configuration and the evolution of nodes' estimates of others' optimal caching decisions.

Since the transport and caching efficiencies and the cache sizes are identical for all nodes, and since the two content pieces are sized to fit within any three caches, we expect that after an initial transient period, nodes 1, 2, and 4 should settle to a caching configuration in which they store approximately equal portions of each content piece, corresponding to their assigned segments (q.v. Fig. 4). If the demands are sufficiently large, we also expect that these three portions together should cover most of each content piece. Moreover, the caching pattern among these nodes should be identical for both content pieces. The same expectations apply to the nodes 2–4, also due to the symmetry of the demand rates and the problem parameters. Fig. 6 shows that the caching configuration at $t = 75$—i.e., the iteration at which the demands change—is consistent with the desired operational principles, and conforms to our expectations.

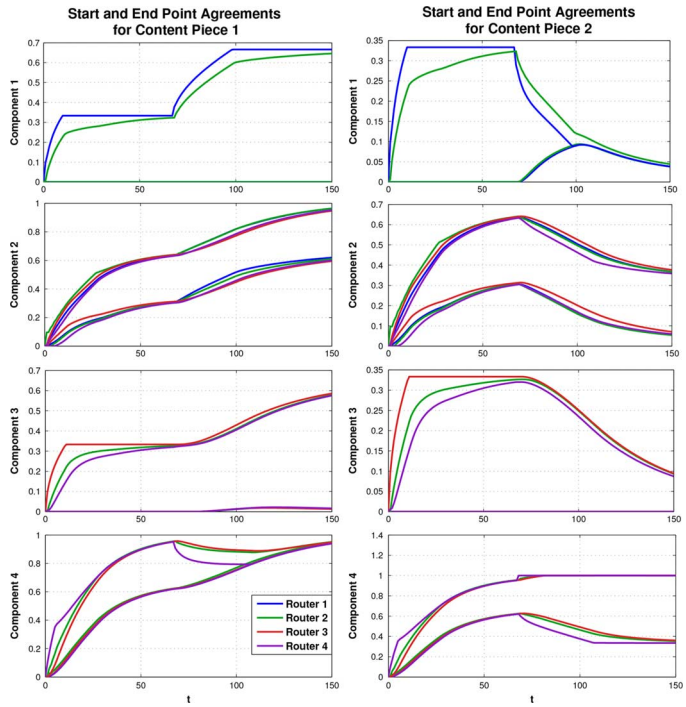


Fig. 7. Time evolution of the content caching configuration. The plots in the first column pertain to content piece 1, while those in the second column pertain to content piece 2. The plots in row i ($i \in \{1, 2, 3, 4\}$) correspond to agents' estimates of x_i^*. Each plot shows the emergence of consensus concerning where an agent's cached block ought to start (bottom set of lines) and where it ought to end (top set of lines).

When demands change at $t = 75$, we expect that node 1 should eventually cache less of content 2 and more of content 1, while node 4 should cache more of content 2 and less of content 1, in accordance with the first operational principle. The content caching configuration at $t = 150$ in Fig. 6 shows that this is indeed the case.

We also note that the caching pattern for node 3 changes from that at $t = 74$, even though the demand that node 3 experiences remains the same throughout the simulation. This happens because $3 \in C_4$; although node 3 does not sense the change in content demand at node 4, its caching decisions are influenced by the "suggestions" that node 4 makes. In cooperating with node 4 (via node 2), node 3 changes its caching behavior so as to help cover those portions of content 1 that node 4 is no longer caching. This behavior is consistent with the third operational principle.

Fig. 6 shows the set of actions implemented by the network nodes at two instants in time, while Fig. 7 shows the complete time evolution of each variable updated in (27), including nodes' "suggestions" to one another. From Fig. 7, we make two observations. First, since nodes' individual costs differ, the nodes are not initially in agreement as to what constitutes the optimal caching configuration. Eventually, however, their estimates reach a consensus. This happens because the first term in the update law (27) is essentially a weighted average of node i's "opinion" and the "opinions" of his neighbors on $\mathcal{G}_j$, on what constitutes j's optimal action, for each $j \in S(i)$. The effect of repeated averaging can thus be intuited as a process by which nodes compromise with one another over individual objectives.

The second observation is that the constraints (4)–(7) are at no time violated throughout the evolution of the algorithm (27).

Remark 6.1 (The Merits of RCO): In implementing the RCO-based update rule (27), the nodes collectively update a total of $2F\sum_{i\in\mathcal{V}}|S(i)|$ real values, and they exchange with their nearest neighbors a total of $2F\sum_{i\in\mathcal{V}}\sum_{j\in S(i)}|\mathcal{N}_j(i)|$ real values at each iteration. By contrast, in implementing an update rule based on the CO algorithm (10), the nodes would need to update $2N^2F$ real values and exchange $2NF\sum_{i\in\mathcal{V}}|\mathcal{N}_C(i)|$ real values at each iteration, where $\mathcal{N}_C(i)$ denotes the set of node i's one-hop neighbors on the communication graph $\mathcal{G}_C$. In a large network with a sparse interference structure, implementing RCO instead of algorithm (10) can result in drastic reductions in communication and processing overhead associated with coordination among the nodes.

Another important benefit achieved by RCO is agents' nescience with regard to the operation of the collective. For example, with the costs defined as in (24), nodes 1 and 3 in this example need not be aware of each others' existence. ♢

Remark 6.2 (The Scalability of RCO): From (22), we observe that the number of real-valued variables updated by node i grows linearly in F and $|S(i)|$, where the set $S(i)$ identifies those subgraphs to which node i belongs. The cardinality of $S(i)$ depends on the way the clusters C_j are assigned to nodes j in the graphical vicinity of node i. Consequently, the maximum number of variables updated by any node within the network is independent of the network size N, and can be minimized by a judicious choice of cluster assignments. On the other hand, since nodes communicate only with their nearest neighbors, the communication overhead associated with coordination among nodes grows with node degree, rather than the size of the network.

The convergence rates of various consensus optimization schemes are studied in [36] (q.v. Lemma 4.6) and several good references therein, including [30] and [32]. Convergence rates of these methods are known to be affected by the properties of the costs $J_i(\cdot)$, as well as the size of the Fiedler eigenvalue $\lambda_2(A)$ (associated with the consensus matrix A in A3.1), whose value is affected by the network connectivity and the choice of link weights [31]. ♢

VII. Performance Evaluation on the COST239

We now develop an RCO-based CCS for the 11-node European optical backbone network COST239, shown in Fig. 8 (q.v. [37] for a more detailed description of this network). We evaluate the performance of our proposed CCS against that of the LFU cache eviction policy along several network-level metrics, which we define in Section VII-B. We describe the simulation setup in Section VII-A, and we discuss the implications of our simulation results in Section VII-C.

A. Simulation Setup

We consider the following problem parameters:

- Router cache sizes: $B_i = 100$ packets, for all $i \in \mathcal{V}$.
- Content catalog: $\mathcal{F} = \{1, \ldots, 20\}$, with $P_k = 50$ packets, for all $k \in \mathcal{F}$.

The uniformity of the cache sizes and content piece sizes is not required for the operation of the RCO-based CCS and is only imposed for convenience. All other parameters are adjusted relative to these, and we find that our ability to tune the cost function and algorithm step size parameters is not affected by the specific choice of catalog, content piece and cache sizes.

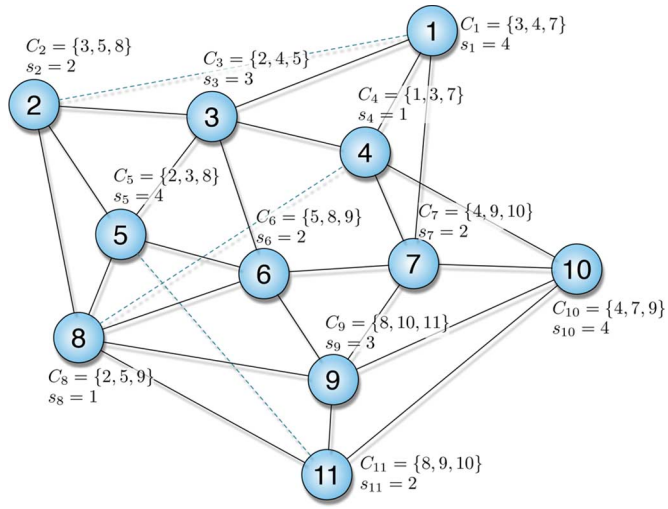


Fig. 8. European optical backbone network, COST239. The segment and cluster assignments used to design the RCO-based CCSs are indicated next to each node. The dashed lines indicate links that are not used to coordinate the nodes.

1) Zipf-Like Demand Rate Signals: From the analysis of several Web proxy traces, it is found in [3]that requests arriving at a Web cache tend to be distributed according to Zipf's law, which states that the relative probability of a request for the rth most popular Web page is proportional to $\frac{1}{r}$. In particular, given a catalog of F pages, with page k being the r_kth most popular page, [3] posits

$$P_F(k) = \frac{\left(\sum_{k=1}^{F}\frac{1}{k^\rho}\right)^{-1}}{r_k^\rho} \tag{31}$$

with typical values of the exponent ρ ranging from 0.6 to 0.8—as a model for the probability that the next request that arrives at a given cache is a request for page k. For exponents in this range, the Zipf-distribution implies that a vast majority of all requests are made only for a small number of the most popular Web pages. Though [3] proposes this Zipf-like model in the context of Web caching, the model is also relevant to caching in the context of CDNs and CCNs [13].

To reflect the features of this model, we generate the demand rate signals according to

$$d_{i,k}(t) = m(t)\cdot\frac{\left(\sum_{k\in\mathcal{F}}\frac{1}{k^{0.75}}\right)^{-1}}{r_k(t)^{0.75}} \tag{32}$$

where $r_k(t)$ is the popularity rank of content piece k at iteration t and $m(t)$ is the magnitude.

To see how the RCO-based CCS recovers from a disturbance, we let $r_k(t) = k$ and $m(t) = 150$ for the first 1000 iterations. For the next 1000 iterations, the popularities of the files are randomly reassigned, but are taken to be the same across all nodes, while $m(t)$ is increased to 195.

2) RCO-Based CCS: We segment each content piece into $M = 4$ parts, and we assign the segments and clusters as indicated next to each node in Fig. 8. Our choice of M is made based on the qualitative observation that most nodes in this network have at least four neighbors. The segments s_i are assigned with

the aim of diversifying the portions of each content piece being cached within the one-hop graphical neighborhood of each node in the network, while the clusters C_i are assigned arbitrarily.

With M, s_i, and C_i selected, the RCO-based CCS takes the form (27), where the cost function $J_i(\cdot)$ is given by (24), the sets X_0 and X_i are given by (18), the link weights $a_{j,k}^{(i)}$ are assigned according to (23), and the caching decision implemented by node i at iteration t is given by $x_i(t) = \xi_{i,i}(t)$, with x_i carrying the meaning in (2). The projection operations in (27) are implemented according to Algorithm 6.1.

The cost function parameters and the algorithm step size α are tunable in our design methodology. After several preliminary runs of the RCO-based CCS specified by our choice of M, s_i, and C_i, we select the following cost function and algorithm parameters in order to obtain an adequate algorithm response, which is qualitatively assessed by means of plots similar to those in Fig. 6.

Cost Function Parameters

- Segmentation boundary terms: $b_{i,j} = 1300$, for all $i \in \mathcal{V}$ and $j \in \{1, 2, 3, 4\}$.
- Coverage terms: $c_i = 500$, for all $i \in \mathcal{V}$.
- Transport efficiencies: $E_{\mathrm{tr},i} = 15$ J per q units of data, for all $i \in \mathcal{V}$ (each data packet is assumed to be q units of data in size)
- Caching efficiencies: $E_{\mathrm{ca},i} = 2$ J per q units of data, for all $i \in \mathcal{V}$.

RCO Parameters

- Step size: $\alpha = 5 \times 10^{-6}$
- Initial conditions: $\xi_{i,j}(0) = [0, \ldots, 0]^T \in \mathbb{R}^{40}$, for all $i \in \mathcal{V}$ and $j \in S(i)$.

A more careful approach to the tuning of the cost function parameters and the step size would likely result in better algorithm performance. We observe that the convergence of this RCO-based CCS does not appear to be sensitive to the choice of initial conditions. Moreover, increasing the magnitude of any of the cost function parameters generally improves algorithm responsiveness in the intended way. For example, increasing $E_{\mathrm{ca},i}$ for all i leads to collective caching decisions that are generally more conservative; a node may not utilize its entire cache at each iteration. On the other hand, increasing the cost function parameters generally necessitates decreasing the algorithm step-size α in order to maintain algorithm stability. However, simulation experience suggests that finding a "sufficiently small" step size is typically not difficult.

Remark 7.1: For this example, an implementation of CO would require 4840 real-valued variables to be updated and 20 240 to be exchanged at each iteration (q.v. Remark 6.1). By contrast, RCO requires 1760 real-valued variables to be updated and only 2640 to be exchanged. $\diamondsuit$

3) LFU Caching: We compare the performance of the RCO-based CCS described in Section VII-A-2 against that of the LFU cache policy, which is known to outperform most local caching policies under the assumption that arriving requests are i.i.d. [3]. LFU caching is particularly effective when content demand follows a Zipf-like distribution [3]. Under LFU, nodes evict the least frequently accessed content in order to make room for new content. We approximate this behavior by a placement policy in which each node caches as much of the most popular content as possible at each t. Since we have chosen $B_i = 2P_k$, for all $i \in \mathcal{V}$ and $k \in \mathcal{F}$, node i implements LFU by caching contents k_1 and k_2, whose demand rate signals $d_{i,k_1}(t)$ and $d_{i,k_2}(t)$ are largest in magnitude at each time t. We allow each node implementing LFU to instantaneously acquire any content piece in response to $d_{i,k}(t)$.

B. Performance Metrics

We compare the performance of the LFU caching policy described in Section VII-A-3 against that of the RCO-based CCS described in Section VII-A-2, along several efficiency-related, network-wide performance metrics. In formulating these metrics, we make the assumption that each node is able to request and acquire individual packets comprising each content piece from the nearest node that caches them.

1) Network Transport Cost (NTC): The NTC is intended to reflect the total actual energy cost associated with delivering uncached content. This metric sums the cost of transporting each uncached packet, in proportion to the rate at which requests arrive for the content piece to which the packet belongs. Specifically, we assume that if a content piece k is requested N_r times at node i, then each packet of k not cached by node i needs to be transported N_r times to node i, from the nearest node in the network that caches the required packet. The cost of transporting each packet is assumed to consist of two components: one that accounts of the energy required for intermediate nodes to route the packet, and another that accounts for the energy expended per unit distance that the packet travels along any network link. We define the NTC as follows:

$$\mathrm{NTC}(t) = \cdot \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{F}} \sum_{p \in \overline{C}_{i,k}(t)} d_{i,k}(t) \cdot (E^{\mathrm{tr},H} H_{i,k,p}(t) + E^{\mathrm{tr},D} D_{i,k,p}(t)) \quad (33)$$

where $\overline{C}_{i,k}(t)$ is the set of all packets comprising content piece k that node i does not cache at time t, $E^{\mathrm{tr},H}$ is amount of energy that a node requires in order to route one packet, $H_{i,k,p}(t)$ is the number of hops that packet p of file k must traverse in order to reach node i from the closest node that caches p at time t, $E^{\mathrm{tr},D}$ is the amount of energy required to transport one packet over a distance of 1 km, and $D_{i,k,p}(t)$ distance that packet p of file k must travel in order to reach node i from the closest node that caches p at time t. The link distances used in this metric are taken from [37]. If node i requires a packet p of content k which is not cached anywhere within the network at time t, then the maximum distance and hop-wise penalties of $D_{i,k,p}(t) =$ 2000 km and $H_{i,k,p}(t) = 10$ are incurred.

2) Average Hops Traveled (AHT) by Uncached Packets: This metric measures the average number of hops that an uncached packet at node i needs to travel in order to arrive at i from the nearest node that caches it. This average is itself then averaged over all nodes in the network. We define the AHT as follows:

$$\mathrm{AHT}(t) = \frac{1}{N} \sum_{i \in \mathcal{V}} \frac{\sum_{k \in \mathcal{F}} d_{i,k}(t) \cdot \left(\sum_{p \in \overline{C}_{i,k}(t)} H_{i,k,p} \right)}{TPRS_i(t)} \quad (34)$$

where $TPRS_i(t)$ is the total number of packet requests sent out by node i at time t—i.e.,

$$TPRS_i(t) = \sum_{i \in \mathcal{F}} d_{i,k}(t) \cdot |\overline{C}_{i,k}(t)|. \quad (35)$$

3) Network Caching Cost (NCC): The NCC reflects the total amount of energy consumed by all the caches in the network. We assume that caching energy is proportional to the amount of content cached [27], and we therefore formulate the NCC metric as

$$\mathrm{NCC}(t) = \sum_{i\in\mathcal{V}} \sum_{k\in\mathcal{F}} E_i^{\mathrm{ca},T} \cdot |C_{i,k}^0(t)| \tag{36}$$

where $E_i^{\mathrm{ca},T}$ is the amount of energy required by node i to cache one packet for T seconds, where T is the duration of one algorithm iteration, and $|C_{i,k}(t)|$ is the number of packets comprising content piece k that node i caches at time t.

4) Node-Averaged Cache Hit Ratios (ACHRs): The *cache hit ratio* (CHR) is a metric that indicates the fraction of requests received that a cache is successfully able to serve. We adapt this notion by considering cache hit ratios within an h-hop radius of a given node. To reflect the network-wide CHR performance, we average these h-hop radius CHRs across all nodes in the network. We define $\mathrm{CHR}_i^h(t)$, the h-hop CHR at node i at time t, as follows. Let $C_{i,k}^h(t)$ be the set of all packets pertaining to content piece $k \in \mathcal{F}$, cached within an h-hop radius of node i at iteration t. That is, $p \in C_{i,k}^h(t)$ implies that at least one node that is at most h hops away from node i is caching packet p. Also, let $TPRR_i(t)$ denote the total number of packet requests received by node i between iteration $t-1$ and t—i.e.,

$$TPRR_i(t) = \sum_{k\in\mathcal{F}} d_{i,k}(t) P_k. \tag{37}$$

Then, the h-hop CHR at node i is given by

$$\mathrm{CHR}_i^h(t) = \frac{\sum_{k\in\mathcal{F}} d_{i,k}(t) \cdot |C_{i,k}^h(t)|}{TPRR_i(t)}, \qquad h = 0, 1, 2, \ldots \tag{38}$$

where $h = 0$ corresponds to the usual notion of CHR at node i, and $h = \infty$ corresponds to a network-wide CHR. Next, we define the Node-Averaged h-hop CHR (ACHR) as

$$\mathrm{ACHR}^h(t) = \frac{1}{N} \sum_{i\in\mathcal{V}} \mathrm{CHR}_i^h(t) \tag{39}$$

and the network-wide, node-averaged CHR as

$$ACHR^{\infty}(t) = \frac{1}{N} \sum_{i\in\mathcal{V}} \frac{\sum_{k\in\mathcal{F}} d_{i,k}(t) \cdot |C_{i,k}^{\infty}(t)|}{TPRR_i(t)}. \tag{40}$$

We assess the performance of the proposed RCO-based CCS described in Section VII-A-2 against that of the LFU caching policy described in Section VII-A-3 along the four CHR metrics $ACHR^0(t)$, $ACHR^1(t)$, $ACHR^2(t)$, and $ACHR^{\infty}(t)$.

The performance of the RCO-based CCS is compared to that of LFU along these metrics, and the results are discussed in the sequel.

C. Results and Implications

The RCO-based CCS and the LFU caching algorithms were run for 2000 iterations. The performance of the RCO-based CCS is compared to that of LFU along the seven metrics defined in Section VII-B, and the results are plotted in Figs. 9 and 10. For simplicity, we assume that all nodes have identical transport efficiency profiles, and we set the related energy efficiency parameters as $E_1^{\mathrm{ca},T} = \cdots = E_N^{\mathrm{ca},T} = E^{\mathrm{tr},D} = E^{\mathrm{tr},H} = 1$.

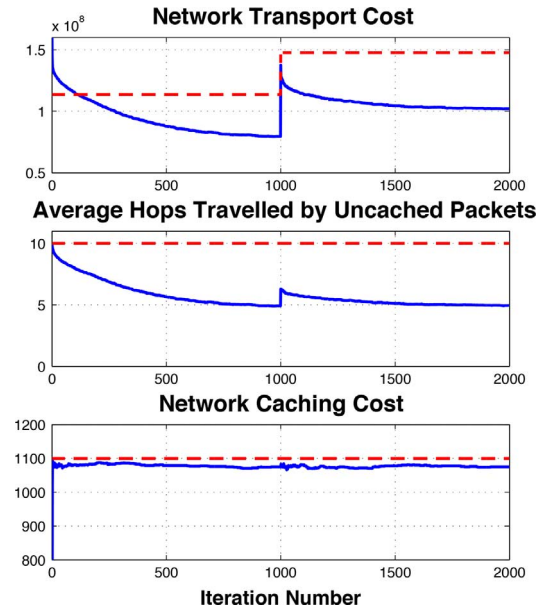


Fig. 9. Performance of the RCO-based CCS (solid lines) compared to that of LFU (dashed lines) along the three metrics calculated using expressions (33) (top plot), (34) (middle plot), and (36) (bottom plot).

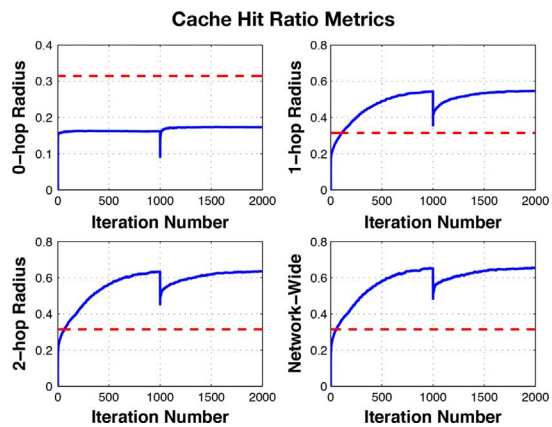


Fig. 10. Performance of the RCO-based CCS (solid lines) compared to that of LFU (dashed lines) along the four Node-averaged CHR metrics $ACHR^0(t)$, $ACHR^1(t)$, $ACHR^2(t)$, and $ACHR^{\infty}(t)$ given by (39) and (40).

Since the RCO-based CCS is initialized with no content cached at any node, its NTC in Fig. 9 is much higher at first. However, in coordinating their caching decisions, nodes eventually achieve a lower network transport cost as indicated by the NTC and AHT plots. The NCC plot indicates that for our choice of cost function parameters, the RCO-based CCS maintains a full cache at each node for most of the time; this may not happen when the cost function parameters are set such that the $E_{\mathrm{tr},i}$ are lower relative to the $E_{\mathrm{ca},i}$.

Even though in both halves of the simulation the LFU caching policy results in each node caching exactly two of the most popular content pieces, the spike in network transport cost observed in the second half of the simulation occurs because the magnitude $m(t)$ of the demand rate signals is increased at $t = 1000$ (q.v. Section VII-A-1). The same spike in the NTC, AHT, and

the four CHR metrics for the RCO-based CCS result additionally from the fact that the caching configuration attained just prior to the change in content popularities imposed at $t = 1000$ is no longer "optimal" once the popularities are changed; there is a transient adjustment period in which algorithm (27) autonomously adapts to the new geographic distribution of content demands.

From Fig. 10, we notice that LFU outperforms the RCO-based CCS for the $ACHR^0(t)$ metric, throughout the entire simulation. This is explained by the fact that LFU caching is especially effective when content demand exhibits a Zipf-like distribution [3]. In our implementation of LFU, each node caches exactly two of the locally most popular content pieces in the catalog, thus ensuring its maximal individual CHR. By contrast, the RCO-based CCS entails what may be interpreted as the nodes' tendency to compromise maximizing their individual performance for the sake of improving network-wide performance. Indeed, the averaged CHR metrics $ACHR^1(t)$, $ACHR^2(t)$, and $ACHR^\infty(t)$ indicate that the network-wide CHR of the RCO-based CCS eventually becomes superior to our implementation of LFU.

It is also interesting to note that the h-hop radius CHRs for this example are not significantly improved for $h > 1$. This is a reflection of our particular choice of cluster and segment assignments indicated in Fig. 8; the clusters are all composed of 1-hop neighbors in this case. In general, clusters may be assigned to include more than a subset of a node's 1-hop neighbors. The price potentially paid is that the number of variables that each node needs to update and exchange with its neighbors at each iteration may increase. The choice of clustering and segmentation represents an important degree of freedom in the design of RCO-based CCSs, and its influence on the efficacy of the resulting CCSs warrants further investigation.

The metrics plotted in Figs. 9 and 10 suggest that RCO-based CCSs may potentially realize significant performance gains relative to caching policies that do not involve coordination among the cache-enabled nodes. Moreover, RCO-based CCSs can exhibit good adaptivity to changes in time-varying problem parameters such as content demand rates.

VIII. Conclusion

In this paper, we developed a flexible methodology for the design of decentralized, adaptive content caching strategies. In response to real-time changes in content demand, a network node implementing the proposed strategy coordinates with its nearest neighbors and updates its caching decisions based on locally available information only. Collectively, the nodes achieve a network caching configuration that best meets a desired set of network-level performance criteria.

We focused on the problem of energy-efficient content delivery over networks with capabilities such as those of CCN. We provided a detailed design example illustrating the application of our methodology, which involves designing node costs in order to balance the tradeoff between energy expended in caching content and that of transporting uncached content.

Our work opens many avenues for future investigation. First, the impact of segment and cluster assignments on network-wide performance metrics has not been investigated. There is a potentially interesting connection between the number of segments (i.e., the number of nodes in a cluster) and the hop-wise distance between content replicas. Previous work that attempts to quantify optimal hop-wise distances between content replicas (q.v. [13], for example) may help inform best practices in making segment and cluster assignments. Second, better designs for the cost functions are certainly possible. For example, one might consider designing these functions in a way that avoids redundant caching among disjoint clusters. Also, though caching and transport are the primary energy-consuming functions within a network, they may not be the only ones; one may consider encoding a more elaborate set of operational principles, which take into account the energy consumed by other network elements. Fourth, although we focused on energy efficiency, other performance objectives can just as easily be incorporated. Fifth, it would be interesting to investigate the algorithms's ability to cope with topological network changes. Specifically, in the present work, cluster and segment assignments are envisioned as initially being assigned offline. Once a network is up and running the proposed CCS, newly added nodes can be assigned segments in a way that maximizes the segment diversity in their graphical neighborhood, and cluster mates in a way that minimizes the node's hop-wise cluster radius.

References

[1] CISCO Systems, Inc., "The zettabyte era (Cisco VNI report)," Technical report, 2012 [Online]. Available: http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights/index.html

[2] S. Michel *et al.*, "Adaptive Web caching: Towards a new global caching architecture," *Comput. Netw. ISDN Syst.*, vol. 30, pp. 2169–2177, 1998.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, 1999, pp. 126–134.

[4] P. Rodriguez, C. Spanner, and E. W. Biersack, "Analysis of Web caching architectures: Hierarchical and distributed caching," *IEEE/ACM Trans. Netw.*, vol. 9, no. 4, pp. 404–418, Aug. 2001.

[5] M. R. Korupolu and M. Dahlin, "Coordinated placement and replacement for large-scale distributed caches," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 6, pp. 1317–1329, Nov.–Dec. 2002.

[6] V. Jacobson *et al.*, "Networking named content," in *Proc. 5th ACM Int. Conf. Emerging Netw. Exper. Technol.*, Rome, Italy, 2009, pp. 1–12.

[7] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. 29th IEEE INFOCOM*, 2010, pp. 1–9.

[8] P. Vixie, "What dns is not," *ACM Queue*, vol. 7, pp. 10–15, 2009.

[9] R. Cuevas, N. Laoutaris, X. Yang, G. Siganos, and P. Rodriguez, "Deep diving into BitTorrent locality," in *Proc. IEEE INFOCOM*, Shanghai, China, 2011, pp. 963–971.

[10] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[11] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. 2nd Ed. ICN Workshop Inf.-Centric Netw.*, 2012, pp. 2920–2931.

[12] U. Lee, I. Rimac, D. Kilper, and V. Hilt, "Toward energy efficient content dissemination," *IEEE Netw.*, vol. 25, no. 2, pp. 14–19, Mar.–Apr. 2011.

[13] K. Guan, G. Atkinson, D. C. Kilper, and E. Gulsen, "On the energy efficiency of content delivery architectures," in *Proc. IEEE ICC*, 2011, pp. 1–6.

[14] N. Choi, K. Guan, D. Kilper, and G. Atkinson, "In-network caching effect on optimal energy consumption in content-centric networking," in *Proc. IEEE ICC Next Generation Netw. Symp.*, 2012, pp. 2889–2894.

[15] J. Llorca *et al.*, "Dynamic in-network caching for energy efficient content delivery," in *Proc. 32nd IEEE INFOCOM*, 2013, pp. 245–249.

[16] D. C. Kilper *et al.*, "Power trends in communication networks," *IEEE J. Sel. Topics Quantum Electron.*, vol. 17, no. 2, pp. 275–285, Mar.–Apr. 2011.

[17] S. Podlipnig and L. Boszormenyi, "A survey of Web cache replacement strategies," *Comput. Surveys*, vol. 35, pp. 374–398, 2003.

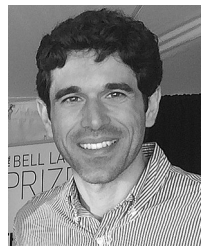
[18] S. Jin and A. Bestavaros, "popularity-aware greedydual-size Web proxy caching algorithms," in *Proc. ICDCS*, 2000, pp. 254–261.

[19] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web caching systems: Modeling, design and experimental results," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1305–1314, Sep. 2002.
[20] A. Jiang and J. Bruck, "Optimal content placement for en-route Web caching," in *Proc. 2nd IEEE Int. Symp. Netw. Comput. Appl.*, 2003, pp. 9–16.
[21] L. Dong, D. Zhang, Y. Zhang, and D. Raychaudhuri, "Optimal caching with content broadcast in cache-and-forward networks," in *Proc. IEEE ICC*, 2011, pp. 1–5.
[22] K. Cho *et al.*, "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," in *Proc. 31st IEEE INFOCOM*, 2011, pp. 316–321.
[23] Z. Li, G. Simon, and A. Gravey, "Caching policies for in-network caching," in *Proc. 21st ICCCN*, 2012, pp. 1–7.
[24] K. Kvaternik, J. Llorca, D. Kilper, and L. Pavel, "Decentralized caching strategies for energy-efficient content delivery," in *Proc. IEEE ICC*, Sydney, Australia, 2014, pp. 3707–3713.
[25] K. Kvaternik, J. Llorca, D. Kilper, and L. Pavel, "A decentralized coordination strategy for networked multiagent systems," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput.*, 2012, pp. 41–47.
[26] K. Kvaternik, "Decentralized coordination control for dynamic multiagent systems," Ph.D. dissertation, University of Toronto, Toronto, ON, Canada, 2015.
[27] L. A. Barroso and U. Holzle, "The case for energy-proportional computing," *IEEE Comput.*, vol. 40, no. 12, pp. 33–37, Dec. 2007.
[28] N. Li and J. R. Marden, "Designing games for distributed optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 2, pp. 230–242, Apr. 2013.
[29] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 1984.
[30] A. Nedić, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.
[31] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, pp. 65–78, 2004.
[32] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–60, Jan. 2009.
[33] A. Vishwanath, J. Zhu, K. Hinton, R. Ayre, and R. Tucker, "Estimating the energy consumption for packet processing, storage and switching in optical-IP routers," in *Proc. Opt. Fiber Commun. Conf.*, 2013, pp. 1–3.
[34] B. Llanas and C. Moreno, "Finding the projection on a polytope: An iterative method," *Comput. Math. Appl.*, vol. 32, pp. 33–39, 1996.
[35] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[36] K. Kvaternik and L. Pavel, "An analytic framework for consensus-decentralized optimization methods: The interconnected systems approach," *IEEE Trans. Autom. Control*, submitted for publication.
[37] G. Rizzelli, A. Morea, M. Tornatore, and O. Rival, "Energy-efficient traffic-aware design of on-off multi-layer translucent optical networks," *Comput. Netw.*, vol. 56, pp. 2443–2455, 2012.

Karla Kvaternik received the B.Sc. degree in electrical and computer engineering from the University of Manitoba, Winnipeg, MB, Canada, in 2006, the M.Sc. degree in control theory from the University of Alberta, Edmonton, AB, Canada, in 2009, and is currently pursuing the Ph.D. degree in control theory from the University of Toronto, Toronto, ON, Canada.

She is currently a Postdoctoral Research Associate with Princeton University, Princeton, NJ, USA, where her research focuses on collective decision making. Her research interests span nonlinear systems and control theory, Lyapunov methods, nonlinear programming, and extremum-seeking control, but her main interest is the development and application of decentralized coordination control strategies for dynamic multiagent systems.

Dr. Kvaternik was the recipient of the prestigious Vanier Canada Graduate Scholarship in 2010 and the recipient of the Best Student Paper award at the 2009 Multiconference on Systems and Control.



Jaime Llorca received the B.S. degree in electrical engineering from the Polytechnic University of Catalonia, Barcelona, Spain, in 2001, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, MD, USA, in 2003 and 2008, respectively.

He has been a Communication Networks Research Scientist with Bell Labs, Holmdel, NJ, USA, since 2010. He held a postdoctoral position with the Center for Networking of Infrastructure Sensors (CNIS), College Park, MD, USA, from 2008 to 2010. He has authored more than 50 peer-reviewed articles and 10 patents. His research interests include energy-efficient networks, distributed cloud networking, content distribution, resource allocation, network information theory, and network optimization.

Dr. Llorca is the recipient of the Best Paper Award at the 2007 International Conference on Sensors, Sensor Networks and Information Processing, Melbourne, Australia, and the 2015 Jimmy H. C. Lin Award for Innovation.

Daniel Kilper (M'07–SM'07) received the Ph.D. degree in physics from the University of Michigan, Ann Arbor, MI, in 1996.

He is a Research Professor with the College of Optical Sciences, University of Arizona, Tucson, AZ, USA, and is currently serving as the Administrative Director for the Center for Integrated Access Networks (CIAN). He holds eight patents and authored four book chapters and more than 100 peer-reviewed publications. Within both academia and industry, he has made contributions in the area of communication devices and networks primarily spanning three areas: energy-efficient communication networks, optical performance monitoring, and dynamic optical networks.

He is an Editor for the Green Communications and Computing Networks Series in *IEEE Communications Magazine* as well as for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Series on Green Communications and Networking. He is a General Chair of IEEE Online Greencomm 2014 and 2015. In addition to his current position, he has served in leadership positions in multi-university/industry consortia including the Center for Telecommunications Value Chain Research (CTVR), Center for Energy Efficient Telecommunications (CEET), and the GreenTouch Consortium. He was the founding Technical Committee Chair of GreenTouch. His work has been recognized with the Bell Labs President's Gold Medal Award, and he served on the Bell Labs President's Advisory Council on Research.

Lacra Pavel (S'93–M'96–SM'04) received the Ph.D. degree in electrical engineering from Queen's University, Kingston, ON, Canada, in 1996.

After a postdoctoral stage with the National Research Council in Ottawa, ON, Canda, and 4 years of working in the industry, she joined the University of Toronto, Toronto, ON, Canada, in 2002, where she is now a Professor with the Department of Electrical and Computer Engineering. Her research is focused on joining game theory, optimization, and control with networks. Her group is working on mathematical algorithms to realize intelligent networks that autonomously self-optimize. She has made specific contributions to algorithms for optical networks and for energy-efficient networks, either in communication or in transportation. Her work resulted in a book, five issued US patents, over 80 publications in top international journals and conferences, and several invited talks.

Prof. Pavel acted as Publications Chair of the 45th IEEE Conference on Decision and Control and has been on the Technical Program Committees of several IEEE conferences.