Distributed Load Management Algorithms in Anycast-based CDNs[☆]

Abhishek Sinha[a,*], Pradeepkumar Mani[b], Jie Liu[d], Ashley Flavel[c,**], Dave Maltz[b]

[a]*Massachusetts Institute of Technology, Cambridge, MA*
[b]*Microsoft Inc., Redmond, WA*
[c]*Salesforce.com, Redmond, WA*
[d]*Microsoft Research, Redmond, WA*

Abstract

Anycast is an internet addressing protocol where multiple hosts share the same IP-address. A popular architecture for modern Content Distribution Networks (CDNs) for geo-replicated services consists of multiple layers of proxy nodes for service and co-located DNS-servers for load-balancing among different proxies. Both the proxies and the DNS-servers use anycast addressing, which offers simplicity of design and high availability of service at the cost of partial loss of routing control. Due to the very nature of anycast, redirection actions by a DNS-server also affects loads at nearby proxies in the network. This makes the problem of optimal distributed load management highly challenging. In this paper, we propose and evaluate an analytical framework to formulate and solve the load-management problem in this context. We consider two distinct algorithms. In the first half of the paper, we pose the load-management problem as a convex optimization problem. Following a Kelly-type dual decomposition technique, we propose a fully-distributed load-management algorithm by introducing *FastControl* packets. This algorithm utilizes the underlying anycast mechanism itself to enable effective coordination among the nodes, thus obviating the need for any external control channel. In the second half of the paper, we consider an alternative greedy

[☆]Part of the paper appeared in 53rd Annual Allerton Conference on Communication, Control and Computing 2015.

[*]Corresponding Author

[**]This work was done when the author was an employee at Microsoft, Redmond.

Email addresses: sinhaa@mit.edu (Abhishek Sinha), prmani@microsoft.com (Pradeepkumar Mani), Jie.Liu@microsoft.com (Jie Liu), aflavel@salesforce.com (Ashley Flavel), dmaltz@microsoft.com (Dave Maltz)

load-management heuristic, currently in production in a major commercial CDN. We study its dynamical characteristics and analytically identify its operational and stability properties. Finally, we critically evaluate both the algorithms and explore their optimality-vs-complexity trade-off using trace-driven simulations.

Keywords: Performance Analysis; Decentralized and Distributed Control; Optimization

---

1. Introduction

With the advent of ubiquitous computing and web-access, the last decade has witnessed an unprecedented growth of Internet-traffic. Popular websites, such as `bing.com`, registers more than a billion hits per day, which need to be processed efficiently in an online fashion, with as little latency as possible. Content Distribution Networks (CDN) are *de facto* architectures to transparently reduce latency between the end-users and the web-services. In CDNs, a collection of non-origin servers (also known as *proxies*) attempt to offload requests from the main servers located in the data-centers, by delivering cached contents on their behalf [13]. Popular examples of internet services using CDN includes web objects, live-streaming media, emails and social networks [23]. Edge-servers with cached contents serve as proxies to intercept some user requests and return contents without a round-trip to the data-centers. Online routing of user-requests to the optimal proxies remains a fundamental challenge in managing modern CDNs. Routing to a remote proxy may introduce extra round-trip delay, whereas routing to an overloaded proxy may cause the request to be dropped.

Anycast is a relatively new paradigm for CDN-management and there are already several commercial CDNs in place today using anycast [8], [6], [26]. With anycast, multiple proxy-servers, having the same user-content, share the same IP-address. Anycast relies on internet-routing protocols (such as, BGP) to route service-requests to any one of the geo-replicated proxies, over a *cheap* network-path [28]. Anycast based mechanisms have the advantage of being simple to deploy and maintain. Being available as a service in IPv6 networks, no global topology or state information are required for its use [2].

Although anycast routing can simplify the system-design and provide a high level of service-availability to the end-users [22], it comes at the cost of partial loss of routing-control. This is because, a user-request may be routed to any one of the multiple geo-replicated proxies having the same anycast address. Since the user-to-proxy routing is done by the Internet routing protocols, which is not

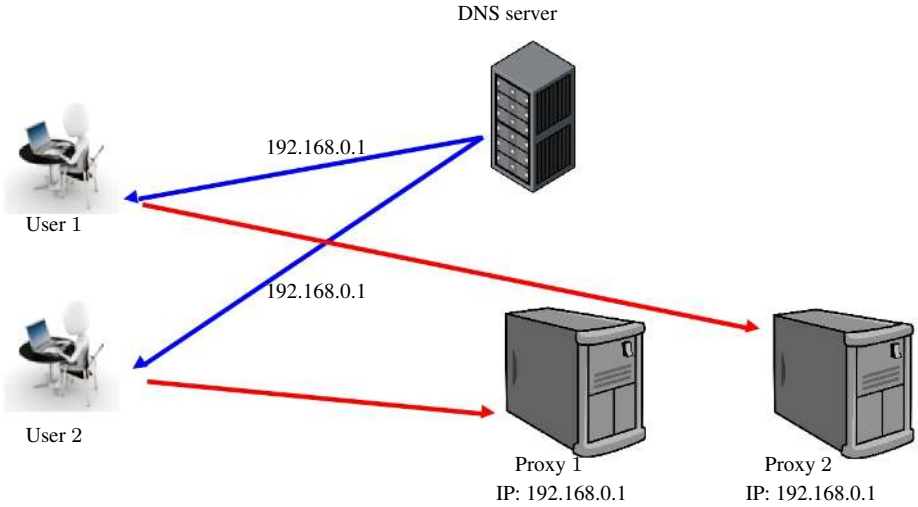

Figure 1: An example of Anycast-enabled CDN. User 1 and 2 obtain the anycast IP-address from the DNS server (blue-dotted arrow) and are then routed over the Internet *either* to Proxy 1 or to Proxy 2 for service (red solid arrow). Note that both proxies have the same IP-address (Anycast addressing). The fraction of users landing on a given proxy is dictated by the Internet routing protocols and is beyond the control of CDN operators. This lack of load-awareness often leads to overloading of certain proxies.

under the control of the CDN-operator, the user-request may end up in an already overloaded proxy, deteriorating its loading-condition further. Figure 1 illustrates the problem of load-management with Anycast.

There have been several attempts in the literature to cope up with the lack of load-awareness issue with network-layer anycast. Papers [16], [9] consider "Active Anycast" where additional intelligence is incorporated into the routers based on RTT and network congestion. It is specifically targeted to reduce pure latency rather than server overload, thus yielding sub-optimal performance in a CDN setting. Alzoubi *et al.* [1] formulates the anycast load-management problem as a General Assignment Problem, which is NP-hard. The paper [10] proposes a new CDN architecture which balances server-load and network-latency via detailed traffic engineering.

In this paper, we focus our attention to a state-of-the-art CDN-architecture, such as *FastRoute* [8], which uses DNS-controlled-redirection for overload control in the proxies. Since DNS (Domain Name System) is the first point-of-contact of users to the internet, DNS-redirection is a popular and effective way to mitigate overload [19], [24]. In this architecture, the proxies are logically arranged in layers of anycast rings, with each layer having a distinct Anycast IP-address.

See Figure 3 for an example. The provisioned capacities of the proxies increase as we move from the outer-layers to the inner-layers of the anycast rings. DNS is responsible for moving load across different layers by intelligently responding to users with different anycast addresses. Each proxy is equipped with a co-located authoritative DNS server. We will collectively refer to a proxy and its co-located DNS-server simply as a *node*. Figure 2 and Figure 3 provide a high-level overview of the proposed CDN-architecture.

An *overload* is said to occur when any proxy receives more service-requests than its processing-capacity. Since DNS is the primary control knob in this architecture, each DNS-server at a node is responsible for redirecting traffic to other layers to alleviate overload in the co-located proxy. A fundamental problem with this approach is that not all users, that hit a given proxy, can be redirected successfully by the co-located DNS. This is because, due to anycast routing, DNS-path and the data-flow paths are independent. Hence, an user's Local DNS (LDNS) could be obtaining a DNS-response from some authoritative DNS-server in a node, which is different from the DNS server co-located with the proxy which the user hits. An example is shown in Figure 2. Hence, intuitively, the ability for a DNS-server at a node to control overload at the corresponding co-located proxy depends on the fraction of oncoming traffic to the proxy that are routed by the co-located DNS-server to the node itself. Informally, we refer to the above quantity as the *self-correlation* [8] of a given node. A formal definition of correlation in this context will be given in Section 2.

Poor self-correlation could impair a node's ability to control overload in isolation. Hence, successful load management in layered CDN should involve coordinated action by DNS-servers in multiple nodes to alleviate overload. Thus the problem reduces to the DNS-plane determining the appropriate offload or redirection probabilities at each node to move traffic from the overloaded proxies to the next layer. This control-decision could be based on variety of information such as load on each proxy, DNS-HTTP correlation etc. From a practical point of view, not all of these quantities are easily measured and communicated to wherever is needed. Thus a centralized solution is not practically feasible and the challenge is to design a provably optimal, yet completely distributed load management algorithm.

Our key contributions in this paper are as follows:

- In section 2, we present a simplified mathematical model for DNS-controlled load-management in modern anycast-based CDNs. Our model is general enough to address the essential operational problems faced by the CDN-

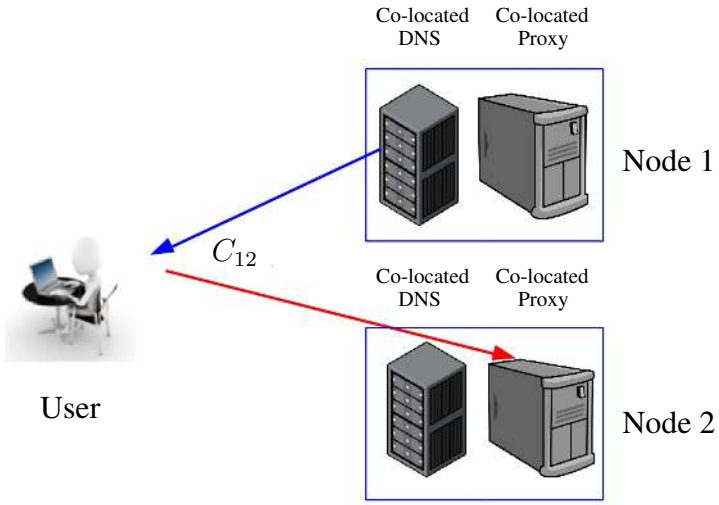

Figure 2: Two *FastRoute* CDN nodes with their corresponding co-located DNS and Proxy servers. Note that, although the user gets served by the proxy at Node 2, it obtains the anycast IP-address from the DNS server at Node 1. Thus, in case of overload, this user can not be redirected from node 2 by altering the DNS-response from the co-located DNS server at node 2.

operators yet tractable enough to draw meaningful analytical conclusions about its operational characteristics.

- In section 3, we formulate the load management problem as a convex optimization problem and propose a Kelly-type *dual* algorithm [11] to solve it in a distributed fashion. The key to our distributed implementation is the *Lagrangian decomposition* and the use of *FastControl* packets, which exploits the underlying anycast architecture to enable coordination among the nodes in a distributed fashion. To the best of our knowledge, this is the first instance of such a decomposition technique employed in the context of load-management in CDNs.

- In section 4, we consider an existing heuristic load-management algorithm used in *FastRoute*, Microsoft's CDN for many first party websites and services it owns [15]. We model the dynamics of this heuristic using non-linear system-theory and analytically derive its operational characteristics, which conform with the qualitative observations. To provide additional insight, a two-node system is analyzed in detail and it is shown, rather surprisingly, that given the "self-correlations" of the nodes are sufficiently high, this heuristic load-management algorithm is able to control an incoming-load of *any* magnitude, however large. Unfortunately, this theoretical guarantee

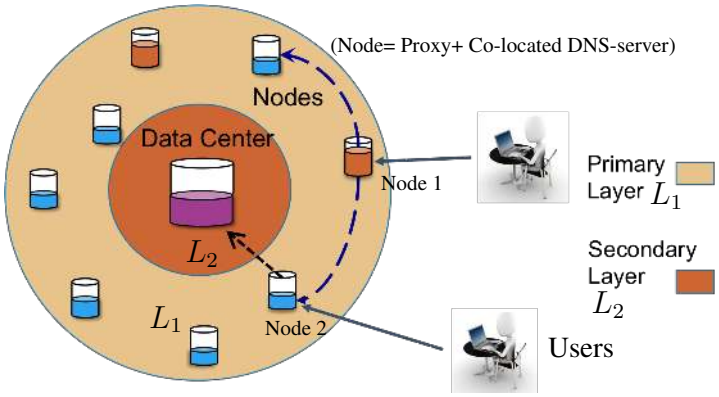

Figure 3: A CDN architecture with two anycast layers. Solid arrows denote user's DNS path. Node 1's DNS returns the anycast address for Layer 1, so, due to anycast, the user gets redirected to other nodes in L_1. Node 2's DNS returns the anycast address for L_2. So the user is redirected to the data-center.

breaks down once this correlation property no longer holds. In this case, the dual algorithm proposed in section 3 performs better than the heuristic.

- In section 5, we critically evaluate relative performance-benefits of the proposed optimal and the heuristic algorithm through extensive numerical simulations. Our simulation is trace-driven in the sense we use real correlation parameters collected over months from a large operational CDN [8].

2. System Model

Nodes and Layers: We consider an anycast-based CDN system consisting of two logical anycast layers: *primary* (also referred to as L_1) and *secondary* (also referred to as L_2). See Figure 3 for a graphic depiction. The primary layer (L_1) hosts a total of N nodes. Each node consists of a co-located authoritative DNS and a proxy server. See Figure 2 for a schematic diagram. The proxy servers are the end-points for user-requests (e.g. HTTP). The i^{th} proxy has a (finite) processing-capacity of T_i user-requests per unit time. The secondary layer (L_2) consists of a single data-center, with practically infinite processing-capacity. Since the proxies in L_1 are geographically distributed throughout the world, an average user is typically located near to at least one proxy, resulting in relatively small user-to-proxy round-trip latency. On the other hand, the single data-center in L_2 is typically located further from the average user-base, resulting in significantly high user-to-data-center round-trip latency. Total response time for a user is the sum of round-

trip latency and processing time of the server (either proxy or the data-center), which we would like to minimize.

Anycast Addressing : As discussed earlier, *all* proxies in the primary layer share the same IP-address $\mathcal{I}_1$ and the data-center in L_2 has a distinct IP-address $\mathcal{I}_2$. This method of multiple proxies sharing the same IP-addresses is known as *anycast* addressing and is widely used for geo-replicated cloud services [27], [1].

Control actions : When a DNS-request by an user [1] arrives at a node i in L_1, requesting for the IP-address of a server, the co-located DNS-server takes *one* of the following actions

- **(1)** It either returns the address $\mathcal{I}_2$ (which offloads the user-request to the data-center) or,
- **(2)** it returns the anycast address $\mathcal{I}_1$ (which, instead, serves the user-request in some proxy in the primary layer L_1 itself).

This (potentially randomized) **binary decision** by the co-located DNS-server of node i could be based on the following local variables measurable at node i :

1. ***DNS-influenced request* arrival rate at node i's co-located DNS-server**, denoted by A_i requests per unit time[2]. Each DNS-request accounts for a certain amount of user-generated load which is routed to L_1 or L_2 according to the DNS-response by node i; We normalize A_i so that each unit of A_i corresponds to a unit of user-load; As an example, if 10% of DNS responses at node i returns the address $\mathcal{I}_2$, then the total amount of load shifted to the data-center (L_2) due to node i's DNS is $0.1A_i$.
2. ***User-load* arrival rate at node i's co-located proxy**, denoted by $S_i(t)$ requests per unit time. This is clearly a function of the DNS-arrivals A_i's and the redirection-decisions of the co-located DNS of different nodes (See Eqn.(2)). The variable $S_i(t)$ is available locally at node i (e.g., HTTP connection-request arrival-rates at the co-located proxy-server, in case of web-applications). We assume that the workloads induced by different user-requests have very small variability (which is true for web search query traffic, e.g.).

[1] In the Internet DNS-requests are actually generated by the Local DNS (LDNS) of the user and are recursively routed to an authoritative DNS server. However, to keep the model and the analysis simple, we will assume that individual DNS-queries are submitted by the users themselves.

[2] We assume that A_i's are piecewise constant and do not change considerably during the transient period of the load-management algorithms discussed here.

Anycasting and inter-node coupling: When a DNS-request arrives at a node i and the co-located DNS-server returns the L_1 anycast-address $\mathcal{I}_1$, the corresponding user-request may be routed to any of one of the N proxy-servers (all having the same IP-address $\mathcal{I}_1$) in the primary layer for service. The particular proxy, where the user-request actually gets routed to, depends on the corresponding ISP's routing policy, network congestion and many other factors. We assume that a typical DNS-query, which is routed to L_1 by the node i, lands at node j's proxy for service with probability C_{ij}, where

$$\sum_{j=1}^{N} C_{ij} = 1, \quad \forall i = 1, 2, \ldots, N \tag{1}$$

In our system, we have determined the matrix $\boldsymbol{C} \equiv [C_{ij}]$ empirically by setting up an experiment similar to the one described in [24], using data collected over several months.

In the ideal case, we would like to have $\boldsymbol{C} \approx \boldsymbol{I}$, where $\boldsymbol{I}$ is the $N \times N$ identity matrix. In this case, the nodes operate in an isolated fashion and situations like Figure 2 rarely takes place. However, as reported in [8], in real CDN-systems with ever-increasing number of proxies, the node are often highly-correlated. In this paper, we investigate the consequences that arise from non-trivial correlations among different nodes and its implications for designing load-management algorithms for CDNs.

System Equations: Assume that, due to action of some control-strategy π, the co-located DNS at node i decides to randomly redirect $1 - x_i^{\pi}(t)$ fraction of incoming DNS-queries (given by A_i) to Layer L_2 at time t ($0 \leq x_i^{\pi}(t) \leq 1$). Thus it routes $x_i^{\pi}(t)$ fraction of the incoming requests to different proxies in the layer L_1. Hence the total user-load arrival rate, $S_i(t)$, at node i's co-located proxy may be written as

$$S_i(t) = \sum_{j=1}^{N} C_{ji} A_j x_j^{\pi}(t), \quad \forall i = 1, 2, \ldots, N \tag{2}$$

A local control strategy π is identified by a collection of mappings $\boldsymbol{\pi} = \Big(x_i^{\pi}(\cdot), i = 1, 2, \ldots, N \Big)$, given by $\boldsymbol{x}_i^{\pi} : \Omega_i^t \times t \to [0, 1]$, where Ω_i^t is the set of all observables at node i up to time t.

3. An Optimization Framework

3.1. Motivation

The central objective of a load-management policy π in a two-layer Anycast-CDN is to route as few requests as possible to the Data-Center L_2 (due to its high round-trip latency), without overloading the primary-layer L_1 proxies (due to their limited capacities). Clearly, these two objectives are at conflict with each other and we need to find a suitable compromise. The added difficulty, which makes the problem fundamentally challenging is that, each node is an autonomous agent and takes its redirection decisions on its own, based on its **local observables only**. As an example, a simple locally-greedy heuristic for node i would be to redirect requests to L_2 (i.e. decrease $x_i(t)$) whenever its co-located proxy is overloaded (i.e., $S_i(t) > T_i$) and redirect requests to L_1 (i.e., increase $x_i(t)$) whenever the co-located proxy is under-loaded (i.e., $S_i(t) < T_i$). This policy forms the basis of the greedy control-strategy used in [8].

This greedy strategy appears to be quite appealing for deployment, due to its extreme simplicity. However, in the next subsection 3.2, we show by a simple example that, in the presence of significantly high cross-correlations among the nodes (i.e. non-negligible value of $C_{ij}, i \neq j$), this simple heuristic could lead to an *uncontrollable overload situation*, an extremely inefficient operating point with degraded service quality. This example will serve as a motivation to come up with a more efficient distributed load-management algorithm, that we develop subsequently.

3.2. Locally Uncontrollable Overload: An Example

Consider a two-layered CDN, hosting only two nodes a and b in the primary layer L_1, as shown in Figure 4. The (normalized) DNS-request arrival rates to the nodes a and b are $A_a = 1$ and $A_b = 1$. Suppose the processing capacities (also referred to as *thresholds*) of the co-located proxies are $T_a = 0.7$ and $T_b = 0.7$ respectively. With the correlation components $[C_{ij}]$ as shown in Figure 4, the user-loads at the co-located proxies are given as

$$S_a(t) = 0.1x_a(t) + 0.5x_b(t) \tag{3}$$

$$S_b(t) = 0.9x_a(t) + 0.5x_b(t) \tag{4}$$

Since $0 \leq x_a(t), x_b(t) \leq 1$, it is clear that under any control policy $\boldsymbol{x}(t)$ the following holds

$$S_a(t) \leq 0.1 \times 1 + 0.5 \times 1 = 0.6 < 0.7 = T_a, \ \forall t$$

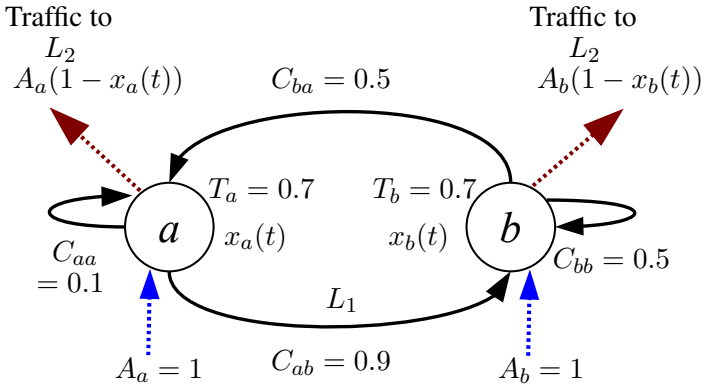

Figure 4: A two-node system illustrating locally uncontrollable overload at the node b

Thus, the proxy at node a will be under-loaded irrespective of the load-management policy π in use. Consequently, under the greedy-heuristic (formally, Algorithm 2 in Section 4), the co-located DNS-server at node a will *greedily* increase its L_1 redirection probability $x_a(t)$ such that $x_a(t) \nearrow 1$ in the steady-state (note that, node a acts autonomously as it does not have node b's loading information). This, in turn, overloads the co-located proxy in node b because the steady-state user-load at proxy b becomes

$$\begin{aligned} S_b(\infty) &= 0.9x_a(\infty) + 0.5x_b(\infty) \\ &= 0.9 \times 1 + 0.5x_b(\infty) \\ &\geq 0.9 > 0.7 = T_b. \end{aligned}$$

Since node b is overloaded in the steady-state, under the action of the above greedy heuristic, it will (unsuccessfully) try to avoid the overload by offloading the incoming DNS-queries to L_2, as much as it can, by letting $x_b(t) \searrow 0$. Thus the steady-state operating point of the algorithm will be $x_a(\infty) = 1, x_b(\infty) = 0$, with node b overloaded. It is interesting to note that poor self-correlation of node a ($C_{aa} = 0.1$) causes the other node b to overload, even under the symmetric DNS-request arrival patterns. Also, this conclusion does not depend on the detailed control-dynamics of the offload probabilities (viz. the instantaneous values if $\dot{x}_1(t)$ and $\dot{x}_2(t)$). Since the overload condition at the node-b can not be overcome by isolated autonomous action of node-b itself, we say that node-b is undergoing a *locally uncontrollable overload situation*.

From the CDN-system point-of-view, the above situation is an extremely inefficient operating-point, because a large fraction (45% in the above example) of the incoming user-requests either gets dropped or severely-delayed due to the overloaded node-b. This poor operating point could have been potentially avoided

provided the nodes somehow mutually co-ordinate their actions[3]. It is not difficult to realize that the principal reasons behind the locally uncontrollable overload situation in the above example are as follows:

- (1) distributed control with local information
- (2) poor self-correlation of node a ($C_{aa} = 0.1$)

The factor (1) is fundamentally related to the distributed nature of the system and requires coordinations among the nodes. In our distributed algorithm [1] we address this issue by introducing the novel idea of *FastControl* packets. This strategy does not require any explicit state or control-information exchange.
Regarding the factor (2), we intuitively expect that the local greedy-heuristic should work well if the self-correlation of the nodes (i.e. C_{ii}) are not too small. In this favorable case, the system will be loosely coupled, so that each node accounts for a major portion of the oncoming load to itself. In section 4, we will return to a variant of this local heuristic used in FastRoute [8] and derive analytical conditions under which the above intuition holds good.
In this section, we take a principled approach and propose an iterative load-management algorithm, which is provably optimal for arbitrary system-parameters $(\boldsymbol{A}, \boldsymbol{C})$. In this algorithm, it is enough for each node i to know its own local DNS and user-load arrival rates (i.e., A_i and $S_i(t)$ respectively) and the entries corresponding to the i^{th} row and column of the correlation matrix $\boldsymbol{C}$ (i.e., $C_{i\cdot}, C_{\cdot i}$). No non-local knowledge of the dynamic loading conditions of other nodes $j \neq i$ is required for its operation.

3.3. Mathematical formulation

We consider the following optimization problem. The variables $\boldsymbol{x}$ and $\boldsymbol{S}$ have the same interpretation as above. The cost-functions, constraints and their connection to the load-management problem are discussed in detail subsequently.

$$\text{Minimize} \qquad W(\boldsymbol{x}, \boldsymbol{S}) \equiv \sum_{i=1}^{N} \big(g_i(S_i) + h_i(x_i)\big) \tag{5}$$

[3]Another trivial solution to avoid overload could be to offload all traffic from all nodes to L_2, i.e. $x_i(t) = 0, \forall i, \forall t$. However, this is highly inefficient because it is tantamount to not using the proxies in the Primary layer L_1 at all.

Subject to,

$$S_i = \sum_{j=1}^{N} C_{ji} A_j x_j, \quad \forall i = 1, 2, \ldots, N \tag{6}$$

$$\boldsymbol{x} \in X, \boldsymbol{S} \in \Sigma_T$$

Discussion. The first component of the cost function, $g_i(S_i)$, denotes the cost for serving S_i amount of user-requests by the i^{th} proxy in L_1 per unit time. Clearly, this cost-component grows rapidly once the proxy is nearly over-loaded, i.e. $S_i \approx T_i$. In our numerical work, we take $g_i(\cdot)$ to be proportional to the average aggregate queuing delay for an $M/G/1$ queue with server-capacity T_i [5], i.e.,

$$g_i(S_i) \;=\; \begin{cases} \frac{\eta_i S_i}{1-\frac{S_i}{T_i}}, & \text{if } S_i \leq T_i \\ \infty, & \text{otherwise} \end{cases} \tag{7}$$

Here η_i is a positive constant, denoting the relative cost per unit increment in latency.
The second component of the cost function $h_i(x_i)$ denotes the cost due to *round-trip-latency* of requests routed to the Data-center (L_2). As an example, in a popular model [20] the delay incurred by a single packet over a congested path varies *affinely* with the offered load. Since the rate of traffic sent to the secondary layer by node i is $A_i(1 - x_i)$, according to this model, the cost-function $h_i(x_i)$ may be taken as follows

$$h_i(x_i) = \theta_i A_i (1 - x_i)\big(d_i + \gamma_i A_i (1 - x_i)\big) \tag{8}$$

where d_i is a (suitably normalized) round-trip-latency parameter from the node i to the data-center in L_2 and θ_i, γ_i are suitable positive constants. We use the cost-functions in Eqns. (7) and (8) for our numerical work. A typical plot of the cost-surface for the case of a two-node system as in Figure 4 is shown in Figure 5.

The constraint set $X = [0, 1]^N$ represents the N-dimensional unit hypercube in which the (controlled) redirection probabilities must lie and the set Σ_T captures

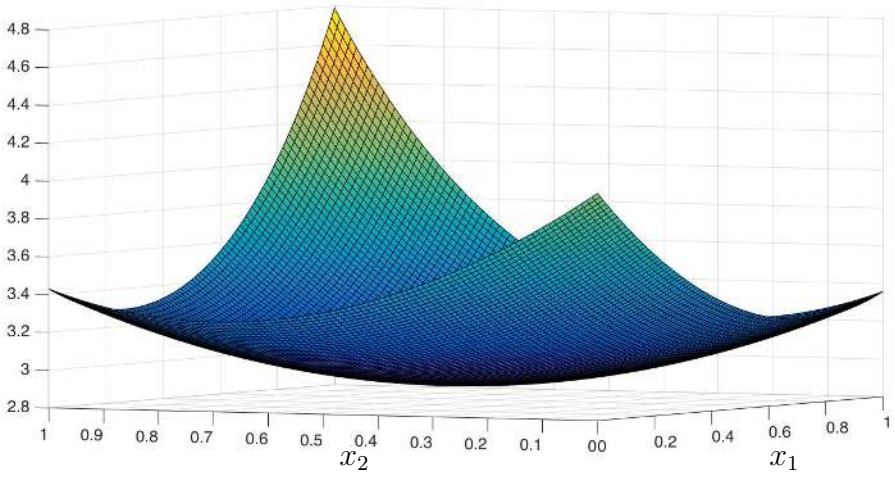

Figure 5: A typical plot of cost-surface for a two-node system as a function of their redirection probabilities $x_1, x_2 (0 \leq x_1, x_2 \leq 1)$.

the capacity constraints of the proxies which are downward closed, e.g., if the proxy i has capacity T_i then we have

$$\Sigma_{\boldsymbol{T}} = \{\boldsymbol{S} : S_i \leq T_i, \forall i = 1, 2, \ldots, N\}$$

The functions $g_i(\cdot), h_i(\cdot)$ are assumed to be closed, proper and strictly convex [3]. We also assume the functions $g_i(\cdot)$ to be monotonically increasing. Hence, we can replace the equality constraint (6) by the following inequality constraint, without loss of optimality

$$\sum_{j=1}^{N} C_{ji} A_j x_j \leq S_i, \quad \forall i = 1, 2, \ldots, N$$

This is because, if the optimal S_i^* is strictly greater than the LHS, we can strictly (and feasibly) reduce the objective value by reducing S_i^* to the level of LHS, resulting in contradiction.
Hence, the above load management problem is equivalent to the following optimization problem $\mathbf{P}_1$:

Minimize $W(\boldsymbol{x}, \boldsymbol{S}) = \sum_{i=1}^{N} \big(g_i(S_i) + h_i(x_i)\big)$
Subject to,

$$\sum_{j=1}^{N} C_{ji} A_j x_j \leq S_i, \quad \forall i = 1, 2, \ldots, N \tag{9}$$

$$\boldsymbol{x} \in X, \boldsymbol{S} \in \Sigma_{\boldsymbol{T}}, \tag{10}$$

where, $X = [0, 1]^N$ and $\Sigma_{\boldsymbol{T}} = \{\boldsymbol{S} : S_i \leq T_i, \forall i = 1, 2, \ldots, N\}$.

Since with the above assumptions, the objective function and the constraint sets of the problem $\boldsymbol{P}_1$ are all convex [3], we immediately have the following lemma:

Lemma 3.1. *The problem $\boldsymbol{P}_1$ is convex.*

Hence a host of methods [4] can be used to solve the problem $\mathbf{P}_1$ in a centralized fashion. However, we are interested in finding a load balancing algorithm for each node, which collectively solve the problem $\mathbf{P}_1$ with locally available loading information only. This problem is explored in the next subsection.

3.4. The Dual Decomposition Algorithm

In this section we derive a dual algorithm [11], [14], [7] for the problem $\mathbf{P}_1$ and show how it leads to a distributed implementation with negligible control overhead.
By associating a non-negative dual variable μ_i to the i^{th} constraint in (9) for all i, the Lagrangian of $\mathbf{P}_1$ may be written as follows:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{S}, \boldsymbol{\mu}) = \sum_{i=1}^{N} \big(g_i(S_i) - \mu_i S_i\big) + \sum_{i=1}^{N} \left(h_i(x_i) + A_i x_i \big(\sum_{j=1}^{N} \mu_j C_{ij}\big)\right) \tag{11}$$

This leads to the following dual objective function [3]

$$D(\boldsymbol{\mu}) = \inf_{\boldsymbol{x} \in X, \boldsymbol{S} \in \Sigma_{\boldsymbol{T}}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{S}, \boldsymbol{\mu}) \tag{12}$$

We now exploit the *separability* property of the dual objective (11) to reduce the problem (12) into following two one-dimensional sub-problems:

$$\left.\begin{aligned} S_i^*(\boldsymbol{\mu}) &= \inf_{0\leq S_i\leq T_i}\Big(g_i(S_i)-\mu_i S_i\Big) \\ x_i^*(\boldsymbol{\mu}) &= \inf_{0\leq x_i\leq 1}\Big(h_i(x_i)+A_i\beta_i(\boldsymbol{\mu})x_i\Big), \end{aligned}\right\} \tag{13}$$

where the scalar $\beta_i(\boldsymbol{\mu})$ is defined as the (scaled) linear projection of the dual-vector $\boldsymbol{\mu}$ on the i^{th} row of the correlation matrix, i.e.

$$\beta_i(\boldsymbol{\mu}) = \sum_{j=1}^{N}\mu_j C_{ij} = \boldsymbol{C}_i^T\boldsymbol{\mu}, \tag{14}$$

and $\boldsymbol{C}_i$ is the i^{th} row of the correlation-matrix $\boldsymbol{C}$.

Discussion. The scalar $\beta_i(\boldsymbol{\mu})$ couples the offload decision of node i with the state of the entire network through Eqn. (13). Once the value of $\beta_i(\boldsymbol{\mu})$ is available to the node i, the node has all the required information at its disposal to *locally* solve the corresponding sub-problems (13), for a fixed value of $\boldsymbol{\mu}$. The solutions of these one-dimensional sub-problems may even be obtained in closed form in some cases. Also, note that $\beta_i(\boldsymbol{\mu})$, being a scalar, is potentially easier to communicate than the entire N dimensional vector $\boldsymbol{\mu}$. In sub-section 3.6, we exploit this fact and show how this factor $\beta_i(\boldsymbol{\mu})$ may be made available to each node i on-the-fly.

With the stated assumptions on the cost functions, there is no duality-gap [3]. Convex duality theory guarantees the existence of an optimal dual variable $\boldsymbol{\mu}^* \geq \mathbf{0}$ such that, the solution to the relaxed problem (13), corresponding to $\boldsymbol{\mu}^*$, gives an optimal solution to the original problem $\mathbf{P}_1$. To obtain the optimal dual variable $\boldsymbol{\mu}^*$, we solve dual $\mathbf{P}_1^*$ of the problem $\mathbf{P}_1$, given as follows

Problem $\mathbf{P}_1^*$:

$$\begin{aligned} \textbf{Maximize} \quad & D(\boldsymbol{\mu}) \\ & \boldsymbol{\mu} \geq \mathbf{0} \end{aligned} \tag{15}$$

The dual problem $\mathbf{P}_1^*$ is well-known to be convex [3]. To solve the problem $\mathbf{P}_1^*$, we use the **Projected Super-gradient** algorithm [17], which will be shown to be amenable to a distributed implementation.
At the k^{th} step of the iteration, a super-gradient $\boldsymbol{g}(\boldsymbol{\mu}(k))$ of the dual function $D(\boldsymbol{\mu})$ at the point $\boldsymbol{\mu} = \boldsymbol{\mu}(k)$ is given by $\partial D(\boldsymbol{\mu}(k)) = \boldsymbol{S}^{\text{obs}}(k) - \boldsymbol{S}(k)$ [3], where $S_i^{\text{obs}}(k)$ is the observed rate of arrival of incoming user-load at the proxy of node i, i.e.,

$$S_i^{\text{obs}}(k) \equiv \sum_{j=1}^{N} C_{ji} A_j x_j^*(k), \tag{16}$$

and $x_i^*(k)$ and $S_i^*(k)$ are the primal variables obtained from Eqn. (13), evaluated at the current dual variable $\boldsymbol{\mu} = \boldsymbol{\mu}(k)$. Following a projected super-gradient step, the dual variables $\boldsymbol{\mu}(k)$ are iteratively updated component-wise at each node i as follows:

$$\mu_i(k+1) = \Big(\mu_i(k) + \alpha\big(S_i^{\text{obs}}(k) - S_i^*(k)\big)\Big)^+ \tag{17}$$

Here α is a small positive step-size constant, whose appropriate value will be given in Theorem (3.3). Since the problem-parameters might vary slowly over time, a stationary algorithm is practically preferable. Hence, we chose to use a constant step-size α, rather than a sequence of diminishing step-sizes $\{\alpha_k\}_{k\geq 1}$. The above constitutes theoretical underpinning of steps (5) and (6) of the proposed distributed load-management algorithm in CDN, described in Algorithm 1.

3.5. Convergence of the Dual Algorithm

To prove the convergence of the above algorithm, we first *uniformly* bound the ℓ_2 norm of the super-gradients $\boldsymbol{g}(\boldsymbol{\mu}(k))$:

$$\boldsymbol{g}(\boldsymbol{\mu}(k)) \equiv \boldsymbol{S}^{\text{obs}}(k) - \boldsymbol{S}(k) = \left(\sum_{j=1}^{N} C_{ji} A_j x_j^*(k) - S_i^*(k)\right)_{i=1}^{N}$$

We start with the following lemma.

Lemma 3.2. *If the total external DNS-request arrival rate to the entire system is bounded by $A_{\max}$ (i.e. $\sum_i A_i \leq A_{\max}$) and the maximum processing-capacity of individual proxies is bounded by $T_{\max}$ (i.e. $T_i \leq T_{\max}, \forall i$) then, for all $k \geq 1$*

$$||\boldsymbol{g}(\boldsymbol{\mu}(k))||_2^2 \leq A_{\max}^2 + NT_{\max}^2 \tag{18}$$

PROOF. See Appendix 7.1.

Upon bounding the super-gradients uniformly for all k, the convergence of the dual algorithm follows directly from Proposition 2.2.2 and 2.2.3 of [4]. In particular, we have the following theorem:

Theorem 3.3. *For a given $\epsilon > 0$, let the step-size α in Eqn.* (17) *be chosen as $\alpha = \frac{2\epsilon}{A_{\max}^2 + NT_{\max}^2}$. Then,*

- *The sequence of solutions, produced by the dual algorithm described above, converge within an ϵ-neighbourhood the optimal objective value of the problem $\boldsymbol{P}_1$.*
- *The rate of convergence of the algorithm to the ϵ-neighborhood of the optima after k-steps is given by $c/\sqrt{k}$ where $c \sim \Theta(\sqrt{N})$.*

The above result states that the rate of convergence of the dual algorithm decreases roughly at the rate of $\Theta(\sqrt{N})$, where N is the total number of nodes in the system. This is expected as more nodes in the system would warrant greater amount of inter-node coordination to converge to the global optimum.

3.6. FASTCONTROL *Packets and Distributed Implementation*

In the previous subsection, we derived a provably optimal load-management algorithm for CDNs, which is implementable online, provided each node i knows how to obtain the value of the coupling-factors $\beta_i(\boldsymbol{\mu}(k))$ at each step, in a decentralized fashion. To accomplish this, we now introduce the novel idea of FASTCONTROL packets. In brief, it exploits the underlying anycast architecture of

the system (Eqn.(2)) for *in-network, on-the-fly computation* of the coupling factor $\beta_i(\boldsymbol{\mu}(k))$ (Eqn. (14)) for all i.

General Descriptions. FASTCONTROL packets are special-purpose control packets (different from the regular data-packets), each belonging to any one of N distinct classes, which we refer to as *categories*. The category of each FASTCONTROL packet is encoded in its packet-header and hence it takes extra $\log(N)$ bits to encode the categories. Physically, these control packets originate from the users and are monitored by the proxies. However, the rates at which these packets are generated, are controlled by the DNS servers of the nodes i and are proportional to the dual variables $\mu_i(k)$, by a mechanism detailed below.

Generation of FastControl packets. These packets are generated in a controlled manner by using a javascript embedded in responses to user DNS-requests (similar to how data was generated to calculate the C matrix offline [8]). The javascript forces users to download a small image from an URL that is not affected by the load management algorithm. DNS-servers in each node are configured to respond back with anycast IP address for the primary layer (i.e. $\mathcal{I}_1$) for this special DNS-query. The use of various categories of FastControl packets will be clear from the description of the following distributed protocol used for determining $\beta_i(\boldsymbol{\mu}(k))$:

- At step k, the DNS server of each node i **forces** generation of FASTCONTROL packets of category j (through its response to DNS-queries) at the rate

$$r_{ij}(k) = \gamma\mu_i(k)\frac{C_{ji}}{C_{ij}}, \quad j = 1, 2, \ldots, N \tag{19}$$

  Note that this step is locally implementable at each node, since the value of the dual variable $\mu_i(k)$ is locally available at the node i. Here $\gamma > 0$ is a fixed system parameter, indicating the rate of control packet generation.

- At each step k, each node i also **monitors** the rate of reception of FASTCONTROL packets of category i, denoted by $R_i(k)$, at its co-located proxy. Using equation (19), the total rate of reception $R_i(k)$ of i^{th} category FASTCONTROL packets at node i is obtained as follows

$$\begin{aligned} R_i(k) &= \sum_{j=1}^{N} r_{ji}(k)C_{ji} = \sum_{j=1}^{N} \gamma\mu_j(k)\frac{C_{ij}}{C_{ji}}C_{ji} \\ &= \gamma\beta_i(\boldsymbol{\mu}(k)) \end{aligned}$$

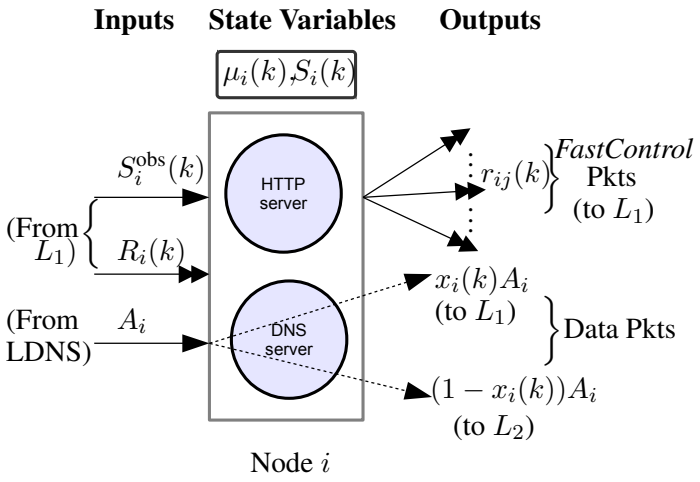

Figure 6: Node-i implementing the dual algorithm

Thus,

$$\beta_i(\boldsymbol{\mu}(k)) = \frac{1}{\gamma} R_i(k) \tag{20}$$

Hence, the value of $\beta_i(\boldsymbol{\mu}(k))$ at node i can be obtained locally by monitoring the rate of receptions of FASTCONTROL packets at the co-located proxy. A complete pseudocode of the algorithm is provided below. See Figure (6) for a schematic diagram of a node implementing the algorithm.

Algorithm 1 Distributed Dual Decomposition Algorithm Running at Node i

1: *Initialize:* $\mu_i(0) \leftarrow 0$
2: **for** $k = 1, 2, 3, \ldots$ **do**
3: **Monitor** $A_i(k), S_i^{\text{obs}}(k), R_i(k)$;
4: Set $\beta_i(k) \leftarrow \frac{1}{\gamma} R_i(k)$;
5: *Update Primal variables:*

$$\begin{aligned} S_i(k) &\leftarrow \inf_{0 \le S_i \le T_i} \big(g_i(S_i) - \mu_i(k) S_i\big) \\ x_i(k) &\leftarrow \inf_{0 \le x_i \le 1} \big(h_i(x_i) + A_i(k)\beta_i(k) x_i\big) \end{aligned}$$

6: *Update Dual variable:*

$$\mu_i(k+1) \leftarrow \big(\mu_i(k) + \alpha(S_i^{\text{obs}}(k) - S_i(k))\big)^+$$

7: Via DNS-response, force users to **generate** FASTCONTROL packets of category j, destined to L_1, at the rate

$$r_{ij}(k+1) = \gamma \mu_i(k+1) \frac{C_{ji}}{C_{ij}}, \quad \forall j = 1, 2, \ldots, N$$

8: For an incoming DNS-query, **respond** with the anycast IP-address for L_1 with probability $x_i(k)$ and IP-address for L_2 with probability $(1 - x_i(k))$.
9: **end for**

We make the following observations :

- The full knowledge of the matrix $\boldsymbol{C}$ at node i is also not necessary. It is sufficient that node i knows the i^{th} row and column of the matrix $\boldsymbol{C}$.
- The optimality of the algorithm **does not depend** on the diagonally dominance property of the correlation matrix $\boldsymbol{C}$, an essential requirement for the greedy heuristic (discussed in the next section) to work reasonably well.
- The parameter γ in Eqn. (19) is directly related to the amount of control-overhead required for the distributed algorithm.

This completes the description of the proposed load-management algorithm, which is completely distributed and provably optimal. In the following section, we concentrate our attention on the load-management heuristic used in FastRoute

and evaluate its performance. We will numerically compare the relative benefits of these two algorithms in Section 5.

4. The Greedy Load-Management Heuristic

In this section, we focus exclusively on the distributed load management heuristic implemented in FastRoute [8], a commercial layered-CDN. This heuristic ignores inter-node correlations altogether. Thus, when an individual proxy becomes overloaded, the co-located DNS-server modifies its DNS-response to redirect more traffic to the data-center (L_2) and vice versa. This simple mechanism is reported to work well in practice when there is high correlation (60-80%) between the node receiving the DNS-query and the node receiving the corresponding user-request. However, in the event of sudden bursts of traffic, e.g., *Flash Crowds*, this greedy heuristic often leads to an uncontrollable overload situation which necessitates manual intervention [8].

Algorithmic challenges: With only local information available at each DNS-server, it faces the following dilemma: offload too little to L_2 and the collocated proxy, if overloaded, remains overloaded; offload too much and the users are directed to remote data-centers and receive an unnecessarily delayed response, due to high round-trip latency. The coupling among the nodes because of inter-node correlation makes this problem highly challenging and gives rise to the so-called uncontrollable overload, discussed earlier in section 3.2.

A pseudo-code showing the general structure of FastRoute's heuristic, running at a node i, is provided below. An explicit control-law modeling the heuristic will be given in section 4.1.

Algorithm 2 Decentralized greedy load-management heuristic used in *FastRoute* [8], running at the node i

for $t = 1, 2, 3, \ldots$ **do**
  if the i^{th} proxy is **under loaded** ($S_i(t) \leq T_i$) **then**
    increase $x_i(t)$ proportional to $-(S_i(t) - T_i)$
  else
    decrease $x_i(t)$ proportional to $(S_i(t) - T_i)$
  end if
end for

Motivation for analyzing the heuristic: The optimal algorithm of section-3 requires the knowledge of the correlation matrix $\boldsymbol{C}$ and needs to utilize addi-

tional FASTCONTROL packets for its operation. In this section we analyze performance of the greedy heuristic, currently implemented in *FastRoute* [8], which does not have these implementation complexities. Since this heuristic completely ignores the inter-node correlations (given by the matrix $\boldsymbol{C}$), it cannot be expected to achieve the optima of the problem $\boldsymbol{P}_1$, in general. Instead, we measure its performance by a coarser performance-metric, given by the number of proxies that undergo uncontrollable overload condition (refer to section 3.2) under its action. Depending on target applications, this metric is often practically good enough for gauging the performance of CDNs.

4.1. Analysis of the Greedy Heuristic

As before, let $x_i(t)$ denote the probability that an incoming DNS-query to the node i at time t is returned with the anycast address of the primary layer L_1. Hence, the total rate of incoming load to the proxy i at time t is given by,

$$S_i(t) = \sum_{j=1}^{N} C_{ji} A_j x_j(t), \quad i = 1, 2, 3, \ldots, N \tag{21}$$

The above system of linear equations can be compactly written as follows

$$\boldsymbol{S}(t) = \boldsymbol{B}\boldsymbol{x}(t) \tag{22}$$

Where,

$$\boldsymbol{B} \equiv \boldsymbol{C}^T \mathbf{diag}(\boldsymbol{A}). \tag{23}$$

Let the vector $\boldsymbol{T}$ denote the processing-capacities (thresholds) of the proxies. As described above, FastRoute's greedy heuristic (2) monitors the overload-metric $S_i(t) - T_i$ and if it is positive (i.e., the node i overloaded), it reduces $x_i(t)$ (i.e., $\frac{dx_i(t)}{dt} < 0$) and if the overload-metric is negative (i.e., the node i under-loaded), it increases $x_i(t)$ (i.e., $\frac{dx_i(t)}{dt} > 0$) proportional to the overload. We consider the following explicit control-law complying with the above general principle:

$$\frac{dx_i(t)}{dt} = -\beta R(x_i(t)) \big(\boldsymbol{B}\boldsymbol{x}(t) - \boldsymbol{T}\big)_i, \quad \forall i \tag{24}$$

The factor $R(x_i(t)) \equiv x_i(t)(1 - x_i(t))$ is a non-negative *damping* component, having the property that $R(0) = R(1) = 0$. This non-linear factor is responsible for restricting the trajectory of $\boldsymbol{x}(t)$ to the N-dimensional unit hypercube $\mathbf{0} \leq$

$\boldsymbol{x}(t) \leq \mathbf{1}$, ensuring the feasibility of the control (24)[4]. The scalar $\beta > 0$ is a sensitivity parameter, relating the robustness of the control-strategy to the local observations at the nodes.
The following theorem establishes soundness of the control (24):

Theorem 4.1. *Consider the following system of ODE*

$$\dot{x_i}(t) = -R(x_i(t))(\boldsymbol{B}\boldsymbol{x}(t) - T)_i, \quad \forall i \tag{25}$$

where $R : [0, 1] \to \mathbb{R}_+$ is any $\mathcal{C}^1$ function, satisfying $R(0) = R(1) = 0$.
Let $\boldsymbol{x}(0) \in$ int$(\mathcal{H})$, where $\mathcal{H}$ is the N-dimensional unit hypercube $[0, 1]^N$. Then the system (25) *admits a unique solution $\boldsymbol{x}(t) \in \mathcal{C}^1$ such that $\boldsymbol{x}(t) \in \mathcal{H}, \forall t \geq 0$.*

PROOF. See Appendix 7.2.

The following theorem reveals an interesting feature of the greedy algorithm, which states that, along any periodic trajectory the average load at any node i is equal to the threshold T_i of that node, and hence they are stable *on the average*.

Theorem 4.2. *Consider the system* (24) *with possibly time-varying arrival rate vector $\boldsymbol{A}(t)$ such that the system operates in a periodic orbit. Then the time-averaged user-load on any node i is equal to the threshold T_i of that node, i.e.*

$$\frac{1}{\tau}\int_0^\tau S_i(t)dt = \bar{T}, \ \ \forall i, \tag{26}$$

where the integral is taken along a periodic orbit of period τ.

PROOF. See Appendix 7.3.

[4]Remember that $x_i(t)$'s, being probabilities, must satisfy $0 \leq x_i(t) \leq 1, \forall t, \forall i$

4.2. Avoiding Locally Uncontrollable Overload

Having established the feasibility and soundness of the control-law (24), we return to the original problem of locally uncontrollable overload, described in Section 3.2. In the following, we derive sufficient conditions for the correlation matrix $\boldsymbol{C}$ and the DNS-request arrival rate vector $\boldsymbol{A}$, for which the system is stable, in the sense that no locally uncontrollable overload situation takes place.

Characterization of the Stability Region

For a fixed correlation matrix $\boldsymbol{C}$, we show that if the arrival rate vector $\boldsymbol{A}$ lies within a certain polytope $\boldsymbol{\Pi_C}$, the system is stable in the above sense, under the action of the greedy load-management heuristic. The formal description and derivation of the result is provided in Appendix 7.4, which involves linearization of the ODE (24) around certain fixed points. Here we outline a simple and intuitive derivation of the stability region $\boldsymbol{\Pi_C}$.

We proceed by contradiction. Suppose that node i is facing a locally uncontrollable overload at time t. Hence, by definition, the following two conditions must be satisfied at node i

$$S_i(\infty) - T_i > 0, \text{ and } x_i(\infty) = 0 \tag{27}$$

Here Eqn. (27) denotes the fact that FastRoute node i is *overloaded*, i.e., the incoming traffic to node i's proxy is more than the capacity of the node i. Eqn. (27) denotes the fact that this overload is *locally uncontrollable*, since even after node i's DNS-server has offloaded *all* incoming DNS-influenced arrivals to L_2 (the best that it can do with its local information), it is facing the overload situation. The above two equations imply that the following condition holds at the node i:

$$\sum_{j \neq i} C_{ji} A_j x_j(\infty) > T_i, \tag{28}$$

where we have used Eqn. (21) and the fact that $x_i(\infty) = 0$. Since $0 \leq x_j(\infty) \leq 1$, a necessary condition for uncontrollable overload (28) at node i is $\sum_{j \neq i} C_{ji} A_j > T_i$. Thus, if $\sum_{j \neq i} C_{ji} A_j \leq T_i$, then the locally uncontrollable overload is avoided at the node i by the greedy heuristic. Taking into account all nodes, we see that if the external DNS-query arrival rate $\boldsymbol{A}$ lies in the polytope $\boldsymbol{\Pi_C}$ defined as

$$\boldsymbol{\Pi_C} = \{\boldsymbol{A} \geq \boldsymbol{0} : \sum_{j \neq i} C_{ji} A_j \leq T_i, \ \forall i = 1, 2, \ldots, N\} \tag{29}$$

then the locally uncontrollable overload situation is avoided at *every* node and the system is stable.
Somewhat surprisingly, by exploiting the exact form of the control-law (24), we can show that a two-node system (as depicted in Figure 4) is able to control DNS-load $\boldsymbol{A}$ of any magnitude, under certain favorable conditions on the correlation matrix $\boldsymbol{C}$.

Special Case

[**Two-node System**]

Consider a two-node CDN discussed earlier in Section 3 (see Figure 4). Let the correlation matrix $\boldsymbol{C}$ for the system be parametrized as follows:

$$\boldsymbol{C}(\alpha,\beta) = \begin{pmatrix} \alpha & 1-\alpha \\ 1-\beta & \beta \end{pmatrix} \tag{30}$$

Then we have the following theorem :

Theorem 4.3. *1) The system does not possess* any *periodic orbit for* any *values of its defining parameters:* $\boldsymbol{A}, \boldsymbol{C}(\alpha,\beta), \boldsymbol{T}$. *Thus the two-node CDN never oscillates.*
2) If $\alpha > \frac{1}{2}$ *and* $\beta > \frac{1}{2}$ *then the system is locally controllable (i.e., no locally uncontrollable overload) for* all *arrival rate-pairs* (A_1, A_2).
3) If $\alpha < \frac{1}{2}$ *and* $\beta < \frac{1}{2}$ *then a sufficient condition for local controllability of the system is* $A_1 < \frac{T_1}{1-\alpha}$, $A_2 < \frac{T_2}{1-\beta}$.

PROOF. The proof of part-(1) follows from Dulac's criterion [25], whereas proof for part-(2) and (3) follows from linearization arguments. See Appendix 7.5 for details.

We emphasize that the part-(2) of the Theorem 4.3 is surprising, as it shows that the system remains locally controllable, no matter how large the incoming DNS-request arrival rate be (c.f. Section 3.2) . The 2D vector-field plot in Figure 7 illustrates the above result. In Figure 4(a), the matrix $\boldsymbol{C}$ is taken to be one satisfying the condition of part (2) of lemma 4.3. As shown, all four phase-plane trajectories with different initial conditions converge to an interior fixed point $(x_1(\infty) > 0, x_2(\infty) > 0)$.

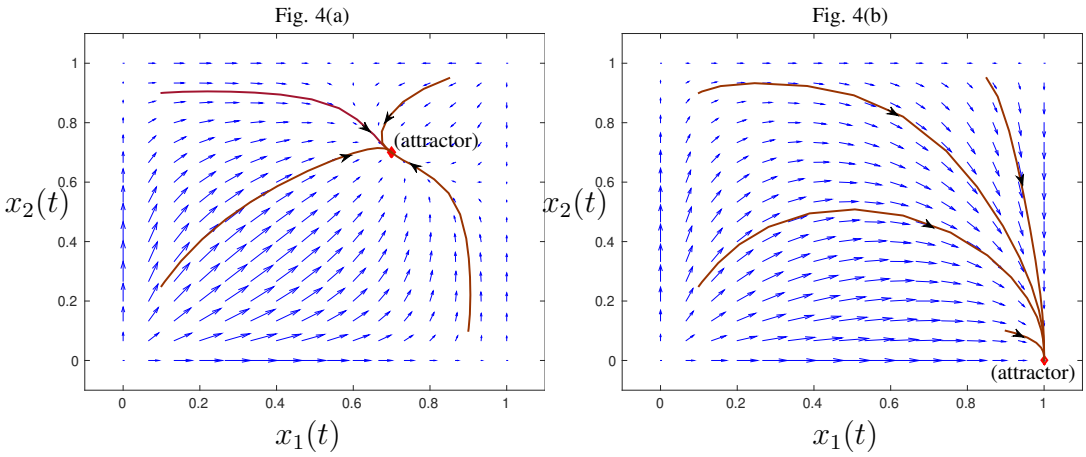

Figure 7: 2D vector-fields of a two-node system illustrating locally controllable (Fig. $4(a)$) and locally uncontrollable (Fig. $4(b)$) overloads.

For the purpose of comparison, in Figure 4(b) we plot the 2D vector-field of a locally uncontrollable system (e.g., the system in Fig. 4). It is observed that all four previous trajectories converge to the uncontrollable attractor $(x_1(\infty) = 1, x_2(\infty) = 0)$. From the vector-field plot, it is also intuitively clear why a periodic orbit can not exist in the system.

Application of Theorem 4.3. Consider a setting where, instead of making locally greedy offloading decisions, nodes are permitted to partially coordinate their actions. Also assume that there exists a partition of the set of nodes into two non-empty disjoint sets G_1 and G_2, such that their effective self-correlation values $\alpha(G_1)$ and $\beta(G_2)$, defined by

$$\alpha(G_1) = \frac{\sum_{i\in G_1}\sum_{j\in G_1} C_{ij}}{|G_1|}, \beta(G_2) = \frac{\sum_{i\in G_2}\sum_{j\in G_2} C_{ij}}{|G_2|}$$

satisfy the condition (2) of Theorem 4.3, i.e. $\alpha(G_1) > \frac{1}{2}, \beta(G_2) > \frac{1}{2}$. Then if the nodes in the sets G_1 and G_2 coordinate and jointly implement the greedy policy, then the system is locally controllable for all symmetric arrivals.

5. Numerical Evaluations

We use the operational FastRoute CDN to identify relevant system parameters for critical evaluations of the optimal algorithm and the heuristic. Currently, FastRoute has many operational nodes, spreading throughout the world [8]. We

show a sample result with $N = 60$ nodes. The inter-node correlation matrix $\boldsymbol{C}$ is computed using system-traces spanning over three months.
For our performance evaluations, we use the cost-functions given in Eqns. (7) and (8). Different (normalized) parameters appearing in the cost-functions are chosen as follows

$$\gamma_i = 1, \theta_i = 10, \eta_i = 1, T_i = 0.7, d_i \sim U_i[0,1] \qquad \forall i \tag{31}$$

where $U_i[0,1]$'s denote i.i.d. uniformly distributed random variables in the range $[0,1]$. The DNS-query arrival rates A_i's are assumed to be i.i.d. distributed according to a Poisson variable with mean $\bar{A}$, which varies across the range $[0.1, 10]$. The optimal solutions of the Lagrangian relaxations in Eqns. (13) for the above cost-functions can be obtained in closed form as follows :

$$S_i^*(\boldsymbol{\mu}) = T_i \max\left(0, 1 - \sqrt{\frac{\eta_i}{\mu_i}}\right) \tag{32}$$

and,

$$x_i^*(\boldsymbol{\mu}) = \begin{cases} 1, & \text{if } c_{2i} > 0 \\ 1 + \frac{c_{2i}}{2c_{1i}}, & \text{if } c_{2i} \leq 0 \text{ and } 2c_{1i} \geq -c_{2i} \\ 0, & \text{o.w.} \end{cases} \tag{33}$$

where $c_{1i} \equiv A_i\theta_i$ and $c_{2i} \equiv \theta_i d_i - \beta_i(\boldsymbol{\mu})$.
For each value of $\bar{A}$, we run the simulation $N_E = 100$ times, by randomizing over both A_i and $d_i, \forall i$. The mean and standard-deviation of the resulting optimal cost values are plotted in the Figure 8(a). As expected, the average cost increases as the overall DNS-query arrivals to the system increases. However, the resulting cost remains finite always. This implies that *none* of the proxies are overloaded.

The above observation is in sharp contrast with the situation using the greedy-heuristic, the subject of Figure 8(b). Here we plot the number of overloaded proxies for different values of $\bar{A}$, keeping all other system-parameters the same. While the greedy algorithm does yield acceptable result for small values of $\bar{A} << T_i = 0.7$, we see that as many as four proxies undergo locally uncontrollable overload for relatively large values of $\bar{A}$.
Thus, depending on the computed correlation-matrix $\boldsymbol{C}$ and a projected bound of the DNS-query arrival rate $\boldsymbol{A}$, we can make an informed decision about the choice of the algorithms to employ in a CDN and the inherent complexity-vs-optimality trade-off it entails.

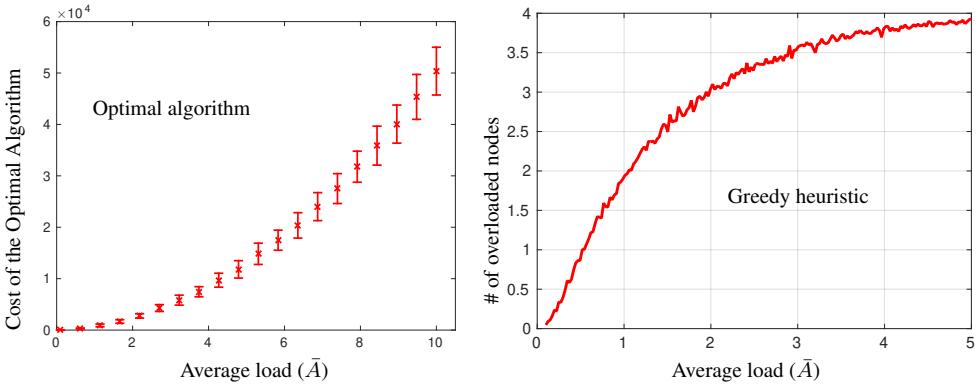

Figure 8: (a) Statistical variation of the cost of the optimal algorithm with mean load (DNS-query arrival rate) $\bar{A}$. (b) Average number of nodes undergoing uncontrollable overload condition under the action of the greedy algorithm (Threshold $T_i = 0.7$ for all nodes). Total number of nodes in the system included in this study is $N = 60$.

6. Conclusion

In this paper we formalize the load management problem in modern CDNs utilizing anycast. We first formulate the problem as a convex optimization problem and study its dual and an associated algorithm. The novel idea of FASTCONTROL packets facilitates distributed implementation of the proposed dual algorithm. Next we analyze stability properties of a greedy heuristic, currently in operation in a major commercial CDN. We find that the optimal algorithm significantly out-performs the heuristic for moderate-to-high value of system load. However, the heuristic performs satisfactorily given that the system is *loosely-coupled* and the offered load is low. Thus an informed choice between these two algorithms may be made depending on the range of system-parameters and the desired optimality/complexity trade-off for a particular CDN. Future work would involve investigating the amount of FASTCONTROL packets necessary for the dual algorithm to work in the presence of random packet-loss and delayed feedback. Also, It would be interesting to generalize the findings of Theorem 4.3 for the case of more than two nodes.

References

References

[1] Alzoubi, H. A., Lee, S., Rabinovich, M., Spatscheck, O., Van der Merwe, J., 2008. Anycast CDNs revisited. In: Proceedings of the 17th International

Conference on World Wide Web. WWW '08. ACM, New York, NY, USA.

[2] Basturk, E., Engel, R., Haas, R., Peris, V., Saha, D., 1997. Using network layer anycast for load distribution in the Internet. In: Tech. Rep., IBM TJ Watson Research Center. Citeseer.

[3] Bertsekas, D. P., 1999. Nonlinear programming. Athena scientific Belmont.

[4] Bertsekas, D. P., 2015. Convex optimization algorithms. Athena Scientific.

[5] Bertsekas, D. P., Gallager, R. G., 1987. Data networks. Prentice-hall.

[6] Engel, R., Peris, V., Saha, D., Basturk, E., Haas, R., 1998. Using IP anycast for load distribution and server location. In: Proc. of IEEE Globecom Global Internet Mini Conference. Citeseer, pp. 27–35.

[7] Eryilmaz, A., Ozdaglar, A., Shah, D., Modiano, E., 2010. Distributed cross-layer algorithms for the optimal control of multihop wireless networks. IEEE/ACM Transactions on Networking (TON) 18 (2), 638–651.

[8] Flavel, A., Mani, P., Maltz, D., Holt, N., Liu, J., Chen, Y., Surmachev, O., 2015. Fastroute: A scalable load-aware anycast routing architecture for modern cdns. In: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). pp. 381–394.

[9] Hashim, H. B., Manan, J.-l. A., 2005. An active anycast RTT-based server selection technique. In: Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 2005 13th IEEE International Conference on. Vol. 1. IEEE, pp. 5–pp.

[10] Jaseemuddin, M., Nanthakumaran, A., Leon-Garcia, A., 2006. TE-friendly content delivery request routing in a CDN. In: Communications, 2006. ICC'06. IEEE International Conference on. Vol. 1. IEEE, pp. 323–330.

[11] Kelly, F., 1997. Charging and rate control for elastic traffic. European transactions on Telecommunications 8 (1), 33–37.

[12] Khalil, H., 1996. Nonlinear systems. 2nd Edition, Prentice Hall.

[13] Krishnamurthy, B., Wills, C., Zhang, Y., 2001. On the use and performance of content distribution networks. In: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement. ACM, pp. 169–182.

[14] Lobel, I., Ozdaglar, A., 2011. Distributed subgradient methods for convex optimization over random networks. Automatic Control, IEEE Transactions on 56 (6), 1291–1306.

[15] Microsoft, 2010. Azure. `http://azure.microsoft.com/en-us/`.

[16] Miura, H., 2001. Server selection policy in active anycast.

[17] Nedic, A., Ozdaglar, A., 2008. Convex optimization in signal processing and communications, chapter cooperative distributed multi-agent optimization. eds., eldar, y. and palomar, d.

[18] Ogata, K., Yang, Y., 1970. Modern control engineering.

[19] Pang, J., Akella, A., Shaikh, A., Krishnamurthy, B., Seshan, S., 2004. On the responsiveness of DNS-based network control. In: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement. ACM, pp. 21–26.

[20] Roughgarden, T., 2005. Selfish routing and the price of anarchy. Vol. 174. MIT press Cambridge.

[21] Rudin, W., 1964. Principles of mathematical analysis. Vol. 3. McGraw-Hill New York.

[22] Sarat, S., Pappas, V., Terzis, A., 2006. On the use of anycast in DNS. In: Computer Communications and Networks, 2006. ICCCN 2006. Proceedings. 15th International Conference on. IEEE, pp. 71–78.

[23] Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., Levy, H. M., 2002. An analysis of internet content delivery systems. ACM SIGOPS Operating Systems Review 36 (SI), 315–327.

[24] Shaikh, A., Tewari, R., Agrawal, M., 2001. On the effectiveness of DNS-based server selection. In: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Vol. 3. IEEE, pp. 1801–1810.

[25] Strogatz, S. H., 2014. Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering. Westview press.

[26] Swildens, E. S.-J., Liu, Z., Day, R. D., Aug. 11 2009. Global traffic management system using IP anycast routing and dynamic load-balancing. US Patent 7,574,499.

[27] Yu, S., Zhou, W., Wu, Y., 2002. Research on network anycast. In: Algorithms and Architectures for Parallel Processing, 2002. Proceedings. Fifth International Conference on. IEEE, pp. 154–161.

[28] Zaumen, W. T., Vutukury, S., Garcia-Luna-Aceves, J., 2000. Load-balanced anycast routing in computer networks. In: Computers and Communications, 2000. Proceedings. ISCC 2000. Fifth IEEE Symposium on. IEEE, pp. 566–574.

7. Appendix

7.1. Proof of Lemma 3.2

PROOF. We have,

$$
\begin{aligned}
||\boldsymbol{g}_k||_2^2 &= \sum_{i=1}^{N}(S_i^{\text{obs}}(k) - S_i(k))^2 && (34)\\
&\leq \sum_{i=1}^{N}\left(S_i^{\text{obs}}(k)\right)^2 + \sum_{i=1}^{N} S_i^2(k) && (35)\\
&\leq \sum_{i=1}^{N}\big(\sum_{j=1}^{N} C_{ji}A_j x_j(k)\big)^2 + NT_{\max}^2 && (36)\\
&\leq \big(\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ji}A_j\big)^2 + NT_{\max}^2 && (37)\\
&= \big(\sum_{j=1}^{N} A_j \sum_{i=1}^{N} C_{ji}\big)^2 + NT_{\max}^2 && (38)\\
&\leq (\sum_{j=1}^{N} A_j)^2 + NT_{\max}^2 && (39)\\
&\leq A_{\max}^2 + NT_{\max}^2 && (40)
\end{aligned}
$$

Here Eqn. (35) follows from non-negativity of S_i^{obs} and S_i, Eqn. (36) follows from the defining equation of S_i^{obs} and the constraint that $S_i \leq T_i$ (viz. Eqn. (13)),

Eqn. (37) follows from the constraint $0 \leq x_i \leq 1, \forall i$, Eqn. (38) follows from the change of order of summation and finally Eqn. (39) follows from the fact that $\boldsymbol{C}$ is a correlation matrix and hence its rows sum to unity (viz. Eqn. (1)).

7.2. Proof of Theorem 4.1

PROOF. First we show that, any solution of the system (24) (if exists) must lie in the unit hypercube $\mathcal{H}$. We prove it via contradiction. On the contrary to the claim, assume that for some solution of the system (24), there exists a component $x_i(\cdot)$ and a finite time $0 \leq \tau < \infty$ such that $x_i(\tau) < 0$. Since $x_i(\cdot)$ is continuous and $x_i(0) > 0$, by *intermidiate value theorem*, there must exist a time $0 < t_0 < \tau$ such that $x_i(t_0) = 0$. Now consider the differential equation corresponding to the i^{th} component of the system (25). We substitute for all other components $\{x_j(t), j \neq i\}$ on the RHS of the following equation.

$$\dot{x_i}(t) = -R(x_i)(\sum_j B_{ij}x_j(t) - T_i), \qquad x_i(t_0) = 0 \tag{41}$$

Since the vector $\boldsymbol{x}(t)$ is $\mathcal{C}^1$ and the regularizer $R(\cdot)$ is assumed to be $\mathcal{C}^1$, the RHS of the equation (41) is $\mathcal{C}^1$. Hence (41) admits a *unique local solution*. However note that the following is a solution to (41)

$$x_i(t) = 0, \qquad \forall t \geq t_0 \tag{42}$$

This is because $R(0) = 0$. By uniqueness, (42) is the *only* solution to (41). This contradicts the fact that $x_i(\tau) < 0$. Hence $\boldsymbol{x}(t) \geq \mathbf{0}, \forall t \geq 0$. In a similar fashion, we can also prove that $\boldsymbol{x}(t) \leq \mathbf{1}, \forall t \geq 0$. This proves that all solutions to (24) must lie in the compact set $\mathcal{H}$.
To complete the proof, we observe that the RHS of the system (24) is locally Lipschitz at each point in the compact set $\mathcal{H}$. Thus the global existence of the solution of (24) follows directly from Theorem 2.4 of [12].

7.3. Proof of Theorem 4.2

PROOF. Let τ be the period of the orbit. Consider the i^{th} differential equation

$$\dot{x_i}(t) = -R(x_i)(S_i(t) - T_i) \tag{43}$$

$$\frac{dx_i}{R(x_i)} = -(S_i(t) - T_i)dt \tag{44}$$

Since $x_i(\cdot)$ belongs to the *interior* of the compact set $\mathcal{H}$, $\frac{1}{R(x_i(t))}$ does not have a zero in the denominator for the enire orbit. Hence $\frac{1}{R(x_i(t))}$ is continuous and its Riemann integral exists [21]. Integrating both sides from 0 to τ, we have

$$\int_0^\tau \frac{dx_i}{R(x_i)} = \int_0^\tau (S_i(t) - T_i)dt \tag{45}$$

Let $J(x_i)$ be an anti-derivative of $\frac{1}{R(x_i)}$. Hence using the fundamental theorem of calculus [21], we can write the LHS of (45) as

$$J(x_i(\tau)) - J(x_i(0)) = \int_0^\tau S_i(t)dt - \tau T_i \tag{46}$$

Since the orbit is assumed to have a period τ, we have $x_i(\tau) = x_i(0)$. Hence $J(x_i(\tau)) - J(x_i(0)) = 0$. Thus we have

$$\bar{S_i} \equiv \frac{1}{\tau}\int_0^\tau S_i(t)dt = T_i$$

7.4. Formal Derivation of the Stability Condition

Let us write the system (24) conveniently as $\dot{\boldsymbol{x}} = \boldsymbol{F}(\boldsymbol{x})$. Consider a fixed point $\overline{x}$ of the system such that the k^{th} node faces an uncontrollable overload condition. By *Hartman-Grobman* theorem [25], it is enough to consider linearized version of the system to determine the stability of fixed points. The first-order Taylor expansion about the fixed point $\overline{x}$ yields the following:

$$\dot{\boldsymbol{x}} \approx F(\bar{\boldsymbol{x}}) + \frac{\partial \boldsymbol{F}(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\overline{\boldsymbol{x}}}(\boldsymbol{x} - \bar{\boldsymbol{x}}) = \frac{\partial \boldsymbol{F}(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\overline{\boldsymbol{x}}}(\boldsymbol{x} - \bar{\boldsymbol{x}})$$

Where, $\frac{\partial \boldsymbol{F}(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\overline{\boldsymbol{x}}}$ denotes the Jacobian [18] of the system (24) evaluated at the fixed point $\boldsymbol{x} = \overline{\boldsymbol{x}}$ and $B_{ij} = C_{ji}A_j$. The second equation follows because $\overline{\boldsymbol{x}}$ is assumed to be a fixed point (and consequently $\boldsymbol{F}(\overline{\boldsymbol{x}}) = \boldsymbol{0}$).
Next we proceed to explicitly compute the Jacobian of the system (24) at a given point $\boldsymbol{x}$. Note that the i^{th} row of the system equation is given by

$$F_i(\boldsymbol{x}) \equiv -x_i(1 - x_i)(\sum_j B_{ij}x_j - T) \tag{47}$$

Thus for $i \neq j$, we have

$$\frac{\partial F_i}{\partial x_j} = -x_i(1 - x_i)B_{ij} \tag{48}$$

and for $i = j$, the diagonal entry is given by

$$\frac{\partial F_i}{\partial x_i} = -\Bigg(x_i(1-x_i)B_{ii} + (1-2x_i)(\sum_j B_{ij}x_j - T) \Bigg)$$
$$= -\Bigg(x_i(1-x_i)B_{ii} + (1-2x_i)(S_i - T) \Bigg) \tag{49}$$

Since the node k is assumed to undergo an uncontrollable overload condition, we necessarily have $x_k = 0$. Hence from Eqns (48) and (49), in the k^{th} row of the Jacobian matrix, the off-diagonal entries are all zero and the diagonal entry is $-(S_k - T)$. Hence if $S_k - T < 0$, at least one eigenvalue of the Jacobian matrix at the fixed point $\overline{x}$ is strictly positive and hence the fixed point $\overline{x}$ is unstable. This implies that a sufficient condition to avoid uncontrollable overload at node k is given by

$$\sum_{j \neq k} C_{jk} A_j \leq T_k \tag{50}$$

where we have used the fact that $x_i(t) \leq 1, \forall i$ and $x_k = 0$. The derivation of the sufficient condtion for absence of uncontrollable overload condition is completed by taking intersection of hyperplanes (50) for all $k = 1, 2, \ldots, N$. ■

7.5. Proof of Theorem 4.3

PROOF. **part-(1) [non-existence of periodic orbits]**
We use Dulac's criterion to prove the non-existence of periodic orbits for the general two-node system. For ease of reference, we recall Dulac's criterion below :

Theorem 7.1 (Dulac's criterion [25]). *Let $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ be a continuously differentiable vector-field defined on a simply connected subset R of the plane. If there exists a continuously differentiable, real-valued function $g(\boldsymbol{x})$ such that $\nabla \cdot (g\dot{\boldsymbol{x}})$ has one sign throughout $\mathcal{D}$, then there are no closed orbits lying entirely in $\mathcal{D}$.*

Now we return to the proof of the result. Let $\mathcal{H}_2 = [0,1]^2$ be the unit square, where by virtue of theorem 4.1, the trajectory of the two-node system lies for all time $t \geq 0$. Thus, it suffices to show that there does not exist any periodic orbit in its interior $\mathring{\mathcal{H}}_2 \equiv \mathcal{D}$. It is obvious that the region $\mathcal{D}$ is simply connected. Now consider the following $g(\boldsymbol{x})$ for application of the Dulac's criterion,

$$g(\boldsymbol{x}) = \frac{1}{x_1 x_2 (1-x_1)(1-x_2)} \tag{51}$$

It is easy to verify that $g(\boldsymbol{x})$ is continuously differentiable throughout the region $\mathcal{D}$. We next evaluate the divergence

$$\nabla \cdot (g\dot{\boldsymbol{x}}) = -\beta\left(\frac{B_{11}}{x_2(1-x_2)} + \frac{B_{22}}{x_1(1-x_1)}\right) \tag{52}$$

It is easy to verify that the term within the parenthesis is always strictly positive throughout the region $\mathcal{D}$. Hence, by Dulac's criterion, there are no closed orbits in $\mathcal{D}$. This proves the result.

part-(2) and (3) [controllability of the system]

Let the arrival rates to node 1 and 2 be given by A_1 and A_2. Let x_1 and x_2 denote the operating point of the system at the steady-state. Our objective is to find sufficient conditions on the arrival rate vector (A_1, A_2), under which the operating points (x_1, x_2) such that either $x_1 = 0$ or $x_2 = 0$ (i.e. full offload to Layer-II) are avoided in the *steady-state*. This will ensure that no uncontrollable overload situation takes place in the system. First we consider the fixed point

$$x_1 = 0, x_2 = 1 \tag{53}$$

This fixed point will be stable if both the eigenvalues of the Jacobian matrix at this point be negative. From Eqns. (48) and (49) we have the following two conditions:

$$S_1 > T, S_2 < T$$

i.e.,

$$(1-\alpha)A_2 > T, \beta A_2 < T$$

i.e.,

$$\frac{T}{1-\alpha} < A_2 < \frac{T}{\beta} \tag{54}$$

Similarly, analyzing the stability of the fixed points around the point $x_1 = 1, x_2 = 0$, we obtain that this fixed point will be unstable if

$$S_1 < T, S_2 > T$$

i.e.,

$$\alpha A_1 < T, (1-\beta)A_1 > T$$

i.e.,

$$\frac{T}{1-\beta} < A_1 < \frac{T}{\alpha} \tag{55}$$

Note that if $\alpha + \beta > 1$, both the above regions (54) and (55) will be empty and the fixed points $(1, 0)$ and $(0, 1)$ will be always unstable. Thus the operating point will not converge to these undesirable fixed points in the steady-state.
Now consider the (possibly feasible) fixed point (x_1, x_2) such that

$$x_1 = 0, S_2 = T \tag{56}$$

The above condition translates to,

$$\begin{aligned} x_1 = 0, A_2 x_2 \beta = T \\ x_1 = 0, x_2 = \frac{T}{A_2 \beta} \end{aligned} \tag{57}$$

This fixed point will be feasible provided $\frac{T}{A_2\beta} < 1$. The Jacobian matrix about this fixed point is evaluated as

$$\frac{\partial \boldsymbol{F}(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{\boldsymbol{x}} = -\begin{pmatrix} (1-\beta)\frac{T}{\beta} - T & 0 \\ \frac{T}{A_2\beta}(1-\frac{T}{A_2\beta})B_{21} & \frac{T}{A_2\beta}(1-\frac{T}{A_2\beta})B_{22} \end{pmatrix}$$

From the Jacobian matrix above, we conclude that both of its eigenvalues are negative provided the following two conditions hold

$$A_2 > \frac{T}{\beta}\,,\ \beta < \frac{1}{2}. \tag{58}$$

Doing similar analysis around the fixed point $(S_1 = T, x_2 = 0)$, we conclude that the above fixed point will be stable for all arrival rates $A_1 > \frac{T}{\alpha}$ and $\alpha < \frac{1}{2}$.
Finally, to obtain efficient operating region for the system (with no uncontrollable overload situation), we take union over stability region of all undesired fixed points and take the complement of it. Hence, if $\alpha > \frac{1}{2}, \beta > \frac{1}{2}$ all the undesrible fixed points are unstable and hence the uncontrollable overload situation is avoided for *all* DNS-request arrival rates $\boldsymbol{A}$. This proves part (2) of the theorem.
On the other hand, if either $\alpha < \frac{1}{2}$ or $\beta < \frac{1}{2}$ holds, then a sufficient condition for the stability of the system is given by

$$A_1 < \frac{T}{1-\alpha}, A_2 < \frac{T}{1-\beta}$$

This proves part (3) of the theorem.