Caching and Neutrality

Muhammad Abdullah
EPFL
Lausanne, Switzerland
muhammad.abdullah@epfl.ch

Pavlos Nikolopoulos
EPFL
Lausanne, Switzerland
pavlos.nikolopoulos@epfl.ch

Katerina Argyraki
EPFL
Lausanne, Switzerland
katerina.argyraki@epfl.ch

Abstract

We are used to defining network neutrality as absence of traffic differentiation, like policing or shaping. These mechanisms, however, are often not what determines end-users' quality of experience (QoE). Most content today is accessed through edge caches, operated by cloud providers, but located near or inside the end-user's Internet Service Provider (ISP). Hence, the end-users' QoE is often determined by the interplay between the caching system (controlled by the cloud provider) and the network between edge cache and end-user (controlled by the eyeball ISP). So, we argue that an obvious point where differentiation may occur, and where transparency and neutrality may be desirable is the caching system; and that we (as a community) should perhaps consider notions of neutrality that capture the connection between caching and QoE.

CCS Concepts

• **Networks** → **Network measurement**; **Network design principles**.

Keywords

Network neutrality, Edge caching

ACM Reference Format:
Muhammad Abdullah, Pavlos Nikolopoulos, and Katerina Argyraki. 2024. Caching and Neutrality. In *Proceedings of The 22nd ACM Workshop on Hot Topics in Networks (HotNets'23).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The point of network neutrality is to enable competition and innovation: the network should not affect the popularity of certain content (e.g., from a particular server or application) by making it easier or harder for end-users to access or enjoy that content; and it should not affect how hard it is for new content to become popular by making it easier to access or enjoy content from established players.

The traditional way in which a network interferes with content is to throttle (or block, at the extreme) the traffic that carries it, and thereby control the rate at which the content reaches each individual end-user and/or the aggregate capacity that it consumes inside the network. Eyeball ISPs notoriously followed this practice with popular peer-to-peer (P2P) applications [6] and still do so with voice and video services (that are not offered by themselves) [10].

Hence, it is not surprising that "neutrality violation" has become synonymous with throttling of target content, implicitly defining network neutrality as absence of selective throttling.

However, we expect that the main reason why end-users today access different content with different performance is not selective throttling—or any kind of differentiation inside the data-plane—but different routes to different content. Today, most content is accessed through the cloud[1]. When an end-user accesses a cloud-hosted piece of content, they may get it from a cloud datacenter, or a cache located at a nearby Point of Presence (PoP), or an "edge cache" located inside their own eyeball ISP. The resulting performance difference can be dramatic.

This situation blurs the line between content carrier and content provider. A cloud certainly provides content (more on this in §2), but it also chooses where (in which ISPs) to place caches, which content (from which particular service or application that it hosts) to cache and where, and which end-users to serve from each cache. An ISP certainly carries content, but it also chooses whose caches to host in its network, and how close to the end-users to place each cache. So, the routes between end-users and content are determined by the interplay between ISPs' and clouds' business and technical goals.

Yet, end-user expectations and Internet regulations have been shaped by a clear separation between the notions of content carrier and content provider. For instance, end-users tend to object to the idea of their ISP inserting ads in their traffic, because they view their ISP as a carrier—and a carrier should never alter the content it carries; yet they tend to accept this practice from clouds, because those are viewed as content

[1]We use the term "cloud" broadly: it may denote a private cloud, a public cloud, or a Content Distribution Network (CDN).

providers. Similarly, we (as a community) tend to object to the idea of an ISP prioritizing some content over other, because a carrier should carry all content equally well; yet it seems natural that a cloud serves some third-party content from right next to the end-user and some other from the other side of the world.

So, we argue that caching should be more transparent. If we care that the network does not affect content popularity and does not make it harder for new content to become popular, then it makes sense to understand the processes that determine which content is served from where.

This paper does not raise any ethical issues.

2 Then and Now

When the notion of network neutrality first appeared in the early 00's, the standard ways to provide an online service were to (a) set up a web server, potentially connected to a back-end, and connect it to the Internet through an edge ISP; or (b) create a peer-to-peer (P2P) application and encourage end-users to install it on their devices. The vision was that, soon, everyone would have access to a good and affordable Internet connection. Hence, to ensure that all online services had more-or-less equal chance of success, the only thing needed was that ISPs did not differentiate across end-users, servers, or applications.

Of course, everyone did *not* have access to a good and affordable Internet connection, not to mention the know-how of setting up web servers or writing P2P applications. Obviously and unavoidably, end-users living in areas with better and cheaper Internet were better-positioned to provide successful online services.

Today, the standard way to provide an online service is to deploy it in the cloud: one's own dedicated private cloud (like Facebook), a public cloud, or both (e.g., Netflix leverages some Amazon services but serves its content through its own CDN). Some public-cloud providers offer edge caching as a service (e.g., Amazon's CloudFront): Suppose a public-cloud customer stores a piece of content in a storage bucket and buys edge caching for it; when an end-user accesses this content (from anywhere in the word), it is served from an edge cache. Some of these edge caches are located in eyeball ISPs, while others are located in the cloud provider's PoPs.

So, providing an online service has arguably become significantly more accessible: At least anyone who can buy basic public-cloud services, no matter where they themselves live and how good their own Internet connection is, can provide an online service that benefits from the reliability and scalability of public clouds—and from their edge caches.

At the same time, network neutrality has become harder to define in a sensible way, in at least three ways:

(1) Many eyeball ISPs host cloud-operated caches ("edge caches") inside their network. The relevant agreements between clouds and ISPs are private, and we don't know whether they involve payment from either side. In a neutral world, an ISP should not prioritize one service's traffic over another's. But isn't it prioritization when one service's content is served from an edge cache while another's is served from a remote data-center? Hence, ideally, edge caching should be done in a way that all online services can benefit from it.

(2) From the point of view of an online service that is hosted in a public cloud (e.g., TikTok or Disney+), the cloud is a content carrier as much as an eyeball ISP is a content carrier for the ISP's end-users. In a neutral world, an ISP should treat equally all end-users who have bought the same ISP service. In the same spirit, a public cloud should treat equally all customers who have bought the same public-cloud service. However, this has implications for the cloud's edge-caching policies: When two customers (say, two social networks, or two video services) buy edge caching for their content, the cloud should ideally serve each customer's content from an edge cache with more-or-less the same probability, even if one customer's content is more popular than the other's. Otherwise, end-users will access the popular customer's content with better performance, even if the two customers are paying for the same public-cloud service.

(3) Cloud providers typically offer their own online services, which may compete with the services of their public-cloud customers. E.g., any video-streaming service deployed on Amazon's public cloud competes with Amazon's Prime Video, and any video-sharing service deployed on Google's public cloud competes with Google's YouTube. This is on par with the common scenario where an ISP offers its own streaming service that competes with third-party streaming services that are accessed by the ISP's end-users. In a neutral world, an ISP should not prioritize traffic from its own service over traffic from third-party services. In the same spirit, a cloud should ideally not serve content that belongs to its own online services from edge caches, while it serves content that belongs to its customers' competing services from remote data-centers, even if the cloud's own online services are more popular.

Today, clouds do not officially disclose their edge-caching policies, and they appear to use different approaches. Some of them appear to use edge caching only for their own content. Others use it both for their own and their customers' content.

It is also interesting to map Facebook's Free Basics[2] program to this landscape: In some countries, Facebook provides free ISP service but limits the online services that its end-users can access to basic health- and education-related websites, as well as the Facebook social network. This is an extreme

[2]https://www.facebook.com/connectivity/solutions/free-basics

example, where: the same entity is an eyeball ISP and a cloud that provides its own online service; the only content that this entity serves to its end-users from within its own network is its own content; and it blocks most third-party content.

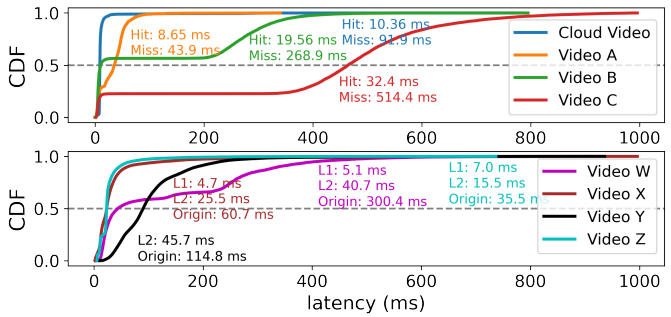
In summary: Clouds and ISPs collaboratively cache some content close to end-users. Content that is served from these edge caches benefits from (potentially significantly) better network performance relative to content that is served from further-away data-centers. Arguably, in a perfectly neutral world (a) all online services would benefit from edge caching, and (b) each online service should experience more-or-less the same hit rate at the same cache. We are absolutely not arguing for regulating clouds' edge-caching policies. But perhaps it is worth defining neutrality in a way that considers edge caching, and monitoring this new kind of neutrality—the way we monitor traffic differentiation by ISPs.

3 Caching (Obviously) Matters

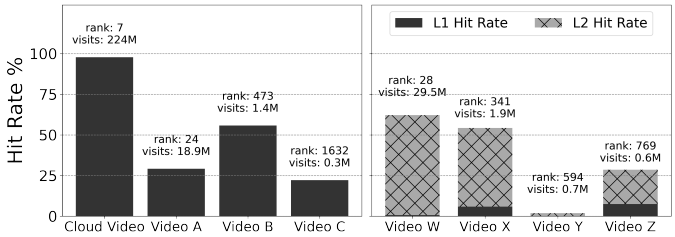
It is expected that caching affects performance, but we wanted to get a sense of magnitude. So, we considered two public clouds (we will call them "Cloud 1" and "Cloud 2"), accessed (from the same location) different online services provided or hosted by them, and compared the resulting end-to-end latency and cache-hit rates. To the best of our understanding, all the customers of each cloud are paying for the same type of edge caching. We start with the results, then briefly comment on the underlying caching architecture and data collection.

Figs. 1 and 2 show the latency distribution and cache-hit rates, experienced by the same client, approximately at the same time, while accessing different online services through edge caches of Cloud 1 or Cloud 2. For Cloud 2, we are able to break down the cache-hit rate into level 1 (L1) and level 2 (L2) hit rate. Some of the latency curves are annotated with the corresponding average hit and miss latency. Each hit-rate bar is annotated with the corresponding service's number of monthly visitors and popularity rank. Fig. 1 concerns video-streaming services. "Cloud Video" is offered by Cloud 1 itself, while all the other video-streaming services are cloud customers, who are paying Cloud 1 or Cloud 2 for edge caching. To the best of our understanding, Cloud 1 uses the same caching infrastructure for its own video-streaming service and its customers' video-streaming services. Fig. 2 concerns websites.

We see significant differences both in terms of latency and in terms of cache-hit rates. Some of these differences can be justified by popularity. E.g., Cloud Video benefits from a significantly higher hit rate than any Cloud-1 customer (98%, while the next highest hit rate is 56%), but it also receives an order of magnitude more monthly visitors than any of them (Fig. 1b). However, this is not the case for all differences. E.g., Video A receives an order of magnitude more visitors than



(a) Latency for services accessed through Cloud 1 (top) and Cloud 2 (bottom).



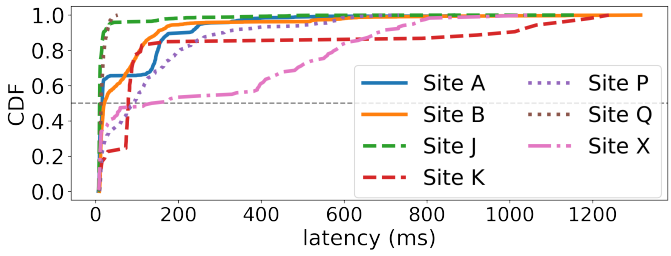
(b) Cache-hit rates for services accessed through Cloud 1 (left) and Cloud 2 (right).

Figure 1: Latency and cache-hit rates for different video-streaming services.

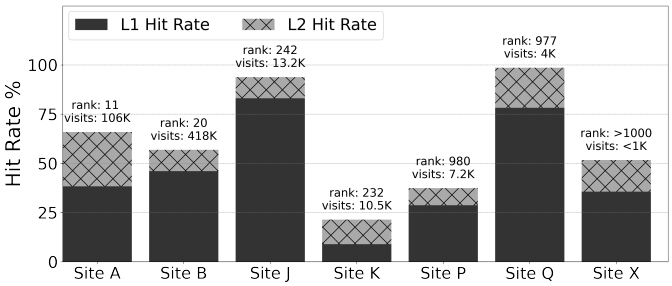
Video B, yet benefits from lower hit rate (Fig. 1b, A: 29%, B:56%). Or, Video Y and Video Z receive a similar number of visitors, yet they benefit from significantly different hit rates (Fig. 1b, Y: 2%, Z: 29%). Or, Sites K, P, and Q receive a decreasing number of visitors, yet benefit from increasing cache-hit rates (Fig. 2b).

Bottom line: The same client accesses similar online services, yet experiences significantly different latency and cache-hit rates, even if the target services are paying for the same edge-caching product. We are not saying that the clouds are doing anything wrong. However, consider the effect on a new and not-yet-popular video-streaming service or website: until it achieves the same popularity as its established competitors, it may be accessed with significantly worse performance.

Caching architecture. By combining publicly available documentation with our measurements, we put together the following picture for the caching architecture of both clouds: The content sits behind a two-tier caching hierarchy, which consists of "user facing" or L1 caches, and "origin-facing" or L2 caches. Some caches are located close to end-users ("edge caches"), while others are located further away ("regional caches"). For all the online services we considered, when a user makes a DNS request for a URL, the DNS response always points to an L1 cache (which is typically an edge cache); if the requested content is not there, it is requested from an L2 cache (which may be an edge or a regional cache);



(a) Latency.



(b) Cache-hit rates.

Figure 2: Latency and cache-hit rates for websites accessed through Cloud 2.

and if it is not there either, it is requested from the content store. As a side note, DNS plays a crucial role, as it determines the edge cache from which each URL is requested, hence, essentially controls the hit rate of each cache.

Data collection. We used a web crawler running on a machine with a stable (wired) Internet connection, located in a city where both clouds have edge caches. To access the video-streaming services: Our crawler played back a few seconds of the top-100 video titles of each service, and it captured the URLs of the first few video chunks; from these, we manually discerned the pattern of video-chunk URLs for each service, and we generated the first 50 URLs for each video title (assuming the first chunks are more likely to be cached). To access websites: We randomly selected 7 from our country's top-1000 websites that cache at least 100 web elements on Cloud 2; for each website, our crawler extracted the URLs of all the web elements (images, JavaScript, CSS, etc.) on the landing page, as well as 8 pages featured on it. For each generated or extracted URL, the crawler sent an HTTP HEAD request and recorded the response time and whether it resulted in a cache hit or miss. Responses from Cloud 1 additionally included the Point of Presence (PoP) from which the URL was served, while responses from Cloud 2 included whether a cache hit was an L1 or an L2 hit. We collected all measurements during the peak hours of activity for the target services.

4 A Neutral-Caching Theory?

We now make a short foray into the world of caching theory to see if it can help us define and reason about the neutrality of modern networks. Neutrality is a kind of fairness, so we revisit the problem of fair caching, focusing on recent work that considers "caching networks," which are meant to model CDNs and/or Information/Content Centric Networks (ICNs and CCNs) [1, 3–5, 8, 11, 13, 14, 17].

4.1 Fair Caching

Fair caching is a utility-optimization problem: given a "caching network," pick a "caching strategy," so as to maximize some notion of "utility." The latter can be a function of cache-hit rate, throughput, latency, routing-cost savings—in general, any metric that caching is supposed to improve.

More formally, a "caching network" is represented as a directed graph $G(V, E)$, where V is a set of caches and E is the set of bidirectional edges with possibly asymmetric routing costs. The network serves content requests for a fixed set of items C as follows: each item $c \in C$ is permanently stored at a subset of "designated" nodes $V_c \subseteq V$, but it may be cached at any node; each request for c, denoted by (c, p), is routed over a pre-established path p that traverses various caches (of perhaps different levels), until it reaches the first node on the path that caches c and can serve the request. The paths do not have loops, and for each request (c, p) only the last node of p is a designated server of c (i.e. the last node belongs to V_c).

A "caching strategy" S typically specifies which content items are cached at each node. It is represented as a $|V| \times |C|$ binary matrix, where each element s_{vc} is equal to 1 if node v caches content c, and 0 otherwise. Existing formulations assume that the admission and eviction policies are the same on all nodes and are given as input; however, it is possible to reformulate the problem to consider eviction and admission policy as part of the strategy, i.e., the output.

As a result of a caching strategy S, each request (c, p) attains a "benefit rate" $z_{(c,p)}(S)$, which could be the achieved cache-hit rate [3–5, 13], throughput [1, 17], latency [7], or caching gain rate (i.e. the reduction in routing costs due to caching) [8, 11, 14].

A "utility function" U is a non-decreasing, continuously differentiable, and strictly concave function that maps the benefit rate to the quantity we want to optimize. The simplest example is $U(x) = x$, where the utility is the benefit rate itself (i.e., we want to optimize the aggregate benefit rate achieved by all requests). Other examples are proportional or max-min fairness. A popular choice for theoretical analysis is the generalized version of an α-fair utility function, which accepts most other types of fairness as special cases.

With these definitions, we can summarize existing formulations of the fair-caching problem as follows:

Traditional Fairness: Find a caching strategy S that maximizes the aggregate utility of all requests per content item, i.e., solve

$$S_{opt} = \arg\max_{S} \sum_{c \in C} U\left(\sum_{(c,p) \in R_c} z_{(c,p)}(S) \right), \quad (1)$$

where the maximization is done across all binary matrices S of size $|V| \times |C|$, and R_c is the set of all requests for content item c.

This formulation achieves "content fairness" but can be easily tweaked to achieve "user fairness" or "content-provider fairness" by changing the summation terms: Instead of optimising the aggregate utility of all requests per content item, we can optimize the aggregate utility of all requests per user, or all requests for content originating from a given content provider.

So, traditional fairness already offers a basis for designing caching networks that treat content, users, and/or content providers fairly, and it admits a flexible definition of fairness. Is anything missing?

4.2 Neutrality as Long-term Fairness

If we define neutrality as a form of *long-term* network behavior, then we must take into account that content popularity may change in response to caching strategy. For example, a strategy that increases the cache-hit rate of a given content item, or decreases the latency with which that item is accessed, may increase the item's popularity over time.

Traditional fairness does not account for this feedback: it assumes that the arrival rate of requests for a given content item is independent from the caching strategy. In particular, most work assumes that the arrival of requests for each content item c, routed over path p, forms a Poisson process with static rate $\lambda_{(c,p)}$. As a result of this assumption, the caching benefit rate $z_{(c,p)}(S)$ is typically computed as the product, $\lambda_{(c,p)} \cdot y_{(c,p)}(S)$, of the corresponding (static) request rate and caching benefit. So, traditional fairness ignores the effect of caching on c's popularity and request rate over time.

We consider two alternative formulations that capture the connection between caching strategy, content popularity, and request rates. In particular, they modify traditional fairness to account for the impact of caching strategy on request rates *over time*.

Horizon Fairness. For each content item c, routed over path p, we assume that the request rate $\lambda_{(c,p)}$ changes over a time horizon H as a static function of the caching benefit $y_{(c,p)}(S)$ achieved through caching strategy S, i.e., $\frac{d\lambda_{(c,p)}}{dt} = f_{c,p}(y_{(c,p)}(S))$. So, Traditional Fairness assumes that request rates are static; our first alternative assumes that request rates are dynamic but change according to a static function $f_{(c,p)}$. Hence, to obtain the caching benefits for any set of (c, p) requests, we integrate over the time horizon H:

Horizon Fairness: Find a caching strategy S that maximizes the aggregate utility of all requests per content item over a given time horizon H, i.e., solve

$$S_{opt} = \arg\max_{S} \sum_{c \in C} U\Bigg(\sum_{(c,p) \in R_c} \frac{1}{H} \int_{t=0}^{H} f_{c,p}(y_{(c,p)}(S)) y_{(c,p)}(S) dt \Bigg), \quad (2)$$

where, as in (1), the maximization is done across all binary matrices S of size $|V| \times |C|$.

Since S is a binary matrix, this is a combinatorial optimization problem, hence NP-hard, even in the case of the trivial utility function $U(x) = x$; however, combinatorial optimization theory offers a large variety of approximation algorithms that can solve it efficiently.

A more interesting challenge is that the function $f_{c,p}$ needs to be known in advance, even for new content items that have never been cached before. One solution would be to use a rough prediction for $f_{c,p}$, based on expert knowledge or experience with similar items. For example, it is plausible that a content provider will have a rough idea, or at least an expectation about how specific caching benefits will affect the popularity of a given content item. In this scenario, solving (2) would achieve fairness with respect to the expected utility of the overall caching benefits per content item. Still, assuming a static function $f_{c,p}$ and requiring knowledge or prediction of that function is arguably non-elegant.

Dynamic Fairness. Our second alternative embraces the fact that a caching network is a dynamic system with feedback: The request rate $\lambda_{(c,p)}$ is not a static function of the caching benefit; instead, it is stochastically affected both by the environment and by the caching strategy. Moreover, we do not seek one caching strategy that is bound to a fixed time horizon H; instead, we seek a dynamic "caching policy" that may apply different caching strategies over time.

A classic tool for modelling such a system is a discrete-time Markov Decision Process (MDP) [16]—a stochastic control process used for decision-making problems, where the outcomes are partially random and partially controllable. More specifically, an MDP is defined by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}_a, \mathcal{R}_a)$, where: $\mathcal{S}$ is a set of states, called "the state space"; $\mathcal{A}$ is the set of actions, called "the action space"; $\mathbb{P}_a$ is the transition probability matrix given some action a—hence, $\mathbb{P}_a(s, s') =$

$\Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that an action a at time t will make the process transition from state s to s'; $\mathcal{R}_a$ is the expected immediate reward received after transitioning from state s to s' as a result of action a.

We map the parameters of our fairness problem to the parameters of the MDP as follows: $\mathcal{S}$ is the set of possible request arrival patterns. In the simplest case, the requests arrive independently from each other, but the rate at which they arrive changes over time; hence, each state s_t is the arrival rate $\lambda_{(c,p)}(t)$ at that point in time. $\mathcal{A}$ contains all possible caching strategies S that can be used at any time t. $\mathcal{R}_a$ are the utilities of the contents achieved due to action a (and the state of the process at the given point in time).

Our goal is to find an optimal caching policy that specifies the action we must choose when the process is in state s. More specifically, we seek a (potentially probabilistic) mapping π from state space $\mathcal{S}$ to action space $\mathcal{A}$ that maximizes the cumulative return (i.e. the expected sum of all content utilities):

Dynamic Fairness: Find a caching policy π that maximizes the aggregate utility of all requests, i.e., solve

$$\pi(s_t) = \arg\max_{a_t} \mathbb{E}\left[\sum_{t+1}^{H} \sum_{c \in C} U\left(\sum_{(c,p) \in R_c} z_{(c,p)}(a_t)\right)\right], \tag{3}$$

where the expectation is taken over all states s_{t+1}, to which the process can transition from state s_t.

We are considering two approaches to solving this problem: One is to assume that the transition probabilities $\mathbb{P}_a$ are known in advance, e.g., through expert knowledge and/or past evidence, in this case, the problem can be solved via classic dynamic programming. The more natural and elegant approach is to assume that the transition probabilities are not known in advance and use (model-free) reinforcement learning [15].

What about admission and replacement? We can easily tweak our formulations to incorporate *some* admission and/or replacement algorithms, e.g., TTL-based replacement, as long as we can compute the caching benefit rate as a function of the value of the timer. This does not apply, however, to algorithms with correlated admission and eviction events, e.g., Least-Recently-Used (LRU) replacement. For such algorithms, one could leverage existing TTL-based approximations [2, 9, 12], but these typically assume static request arrival models—hence need to be extended to our dynamic framework. In summary, going from content placement to admission and replacement seems plausible, yet challenging.

5 Discussion

"No special deals" as opposed to optimal utility. The last section defined neutrality as optimal long-term utility; a simpler alternative is "charging for service/resources in a uniform/public fashion, i.e. no special deals for anyone," as one of our reviewers put it. Under this definition, a caching provider is neutral as long as it charges all customers the same for the same type of caching *and* it applies customer-agnostic caching strategies (i.e., content placement and admission/eviction algorithms). The fact that we explored a utility-based definition does not mean that we are arguing against this alternative. We are arguing that (a) it would be useful to have *a* reasonable definition; and (b) public clouds and ISPs should make their caching strategies transparent, such that one can reason about their neutrality.

Not regulation; transparency and monitoring. We are not advocating for regulating caching strategies; only for making them transparent, such that end-users and cloud customers can understand how caching affects, respectively, their QoE and business. To confirm that a caching provider follows a strategy it claims to follow, one needs to track the popularity of the cached items, at a finer granularity than the number of monthly visitors (which is what is typically disclosed today). Each caching customer obviously tracks the popularity of its own items; an interesting question is how much information they need to exchange in order to collaboratively confirm their caching provider's claimed strategy.

Do we really need a definition? One might argue that we don't need a formal definition of neutrality: Performance differences don't matter, as long as content from all online services is accessed with good-enough performance, where "good enough" depends on the application. If end-users care about neutrality, they should simply monitor the performance with which they access competing online services; if they find the differences significant enough to affect their QoE, they should complain about it. In a competitive world, and as long as enough end-users care, some cloud provider will offer the desired behavior in order to attract more customers. This is a plausible path to neutrality—as long as the cloud world remains competitive, and as long as switching public-cloud providers remains a viable option for online services. Then again, one might argue that a formal definition can only help: If end-users can access all online services with good-enough performance, and performance differences don't significantly affect QoE and service popularity, then a correct definition will capture this reality and give us assurance that the Internet is and will remain neutral.

Acknowledgments

We would like to thank the HotNets reviewers for their insightful feedback.

References

[1] Thomas Bonald, Léonce Mekinda, and Luca Muscariello. 2017. Fair throughput allocation in Information-Centric Networks. *Computer Networks* 125 (2017), 122–131. https://doi.org/10.1016/j.comnet.2017.05.019 Softwarization and Caching in NGN.

[2] Hao Che, Ye Tung, and Zhijun Wang. 2002. Hierarchical Web caching systems: modeling, design and experimental results. *IEEE Journal on Selected Areas in Communications* 20, 7 (2002), 1305–1314. https://doi.org/10.1109/JSAC.2002.801752

[3] Weibo Chu, Mostafa Dehghan, John C.S. Lui, Don Towsley, and Zhi-Li Zhang. 2018. Joint cache resource allocation and request routing for in-network caching services. *Computer Networks* 131 (2018), 1–14. https://doi.org/10.1016/j.comnet.2017.11.009

[4] Mostafa Dehghan, Weibo Chu, Philippe Nain, Don Towsley, and Zhi-Li Zhang. 2019. Sharing Cache Resources Among Content Providers: A Utility-Based Approach. *IEEE/ACM Transactions on Networking* 27, 2 (2019), 477–490. https://doi.org/10.1109/TNET.2018.2890512

[5] Mostafa Dehghan, Laurent Massoulié, Don Towsley, Daniel Sadoc Menasché, and Y. C. Tay. 2019. A Utility Optimization Approach to Network Cache Design. *IEEE/ACM Transactions on Networking* 27, 3 (2019), 1013–1027. https://doi.org/10.1109/TNET.2019.2913677

[6] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, Krishna P. Gummadi, Ratul Mahajan, and Stefan Saroiu. 2010. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*. USENIX Association, USA, 27.

[7] Guilherme Domingues, Edmundo de Souza e Silva, Rosa M.M. Leão, Daniel S. Menasché, and Don Towsley. 2017. Enabling opportunistic search and placement in cache networks. *Computer Networks* 119 (2017), 17–34. https://doi.org/10.1016/j.comnet.2017.03.005

[8] Stratis Ioannidis and Edmund Yeh. 2016. Adaptive Caching Networks with Optimality Guarantees. *ACM SIGMETRICS Perform. Eval. Rev.* 44, 1 (jun 2016), 113–124. https://doi.org/10.1145/2964791.2901467

[9] Bo Jiang, Philippe Nain, and Don Towsley. 2018. On the Convergence of the TTL Approximation for an LRU Cache under Independent Stationary Request Processes. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3, 4, Article 20 (sep 2018), 31 pages. https://doi.org/10.1145/3239164

[10] Fangfan Li, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. 2019. A Large-Scale Analysis of Deployed Traffic Differentiation Practices. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 130–144. https://doi.org/10.1145/3341302.3342092

[11] Yuezhou Liu, Yuanyuan Li, Qian Ma, Stratis Ioannidis, and Edmund Yeh. 2021. Fair Caching Networks. *ACM SIGMETRICS Perform. Eval. Rev.* 48, 3 (mar 2021), 89–90. https://doi.org/10.1145/3453953.3453973

[12] Valentina Martina, Michele Garetto, and Emilio Leonardi. 2014. A unified approach to the performance analysis of caching systems. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 2040–2048. https://doi.org/10.1109/INFOCOM.2014.6848145

[13] Nitish Panigrahy, Jian Li, Faheem Zafari, Don Towsley, and Paul Yu. 2021. *A TTL-based Approach for Content Placement in Edge Networks*. 1–21. https://doi.org/10.1007/978-3-030-92511-6_1

[14] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G. Dimakis, Andreas F. Molisch, and Giuseppe Caire. 2013. FemtoCaching: Wireless Content Delivery Through Distributed Caching Helpers. *IEEE Transactions on Information Theory* 59, 12 (2013), 8402–8413. https://doi.org/10.1109/TIT.2013.2281606

[15] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

[16] William Uther. 2010. *Markov Decision Processes*. Springer US, Boston, MA, 642–646. https://doi.org/10.1007/978-0-387-30164-8_512

[17] Edmund Yeh, Tracey Ho, Ying Cui, Michael Burd, Ran Liu, and Derek Leong. 2014. VIP: A Framework for Joint Dynamic Forwarding and Caching in Named Data Networks. In *Proceedings of the 1st ACM Conference on Information-Centric Networking (ACM-ICN '14)*. Association for Computing Machinery, New York, NY, USA, 117–126. https://doi.org/10.1145/2660129.2660151