Carbon-Aware Name Resolution

Ryan Gibb
University of Cambridge
ryan.gibb@cl.cam.ac.uk

Patrick Ferris
University of Cambridge
patrick.ferris@cl.cam.ac.uk

Anil Madhavapeddy
University of Cambridge
anil.madhavapeddy@cl.cam.ac.uk

ABSTRACT

The current Internet architecture fails to treat the carbon emissions associated with networked services as a first-class metric. We propose extending the DNS with load balancing techniques to consider the carbon cost of scheduling decisions, and further to actively wake machines running networked services as a side effect of name resolution. By extending the DNS, we maintain compatibility with existing Internet infrastructure, unlocking the ability for existing applications to be carbon-aware.

1 INTRODUCTION

In a world in the midst of unprecedented climate change, we should minimise the carbon usage of our networked services, both local and global. To address this we (1) extend the existing mechanism of DNS load balancing with carbon-awareness by creating a carbon scheduling algorithm, and (2) we *actively* manage the power state of machines by powering them down when idle and waking them as a side effect of a name resolution, trading latency and availability for energy efficiency. Leveraging the DNS to implement these policies allows them to interoperate with Internet protocols, unlocking carbon awareness for existing applications.

2 CARBON-AWARE LOAD BALANCING

DNS load balancing is an established way of varying the results returned by a DNS query according to a policy, often round-robin scheduling [6]. Routing and load balancing decisions can also be made in order to (1) minimise latency by providing IP addresses topologically and geographically close to the requester, (2) maximize availability by taking machines out of rotation, and (3) horizontally scale services beyond what a single machine could handle by using multiple machines. Carbon-aware load balancing policies are well established [11, 13], but are typically realized in the application layer which hinders their ease of deployment for existing services. We propose implementing load balancing policies using DNS as the mechanism through which domain-specific policies can be enforced.

An example of such a carbon-aware load balancing policy is heliotropic resolution – following the sun. The *Solar Protocol* offers the ability to host a particular service across geospatially disparate, solar-powered servers and update DNS record entries based on local server information like sunshine and battery capacity [5]. However, it exists at the application layer requiring a bespoke network of solar-powered machines to host Internet services with each server updating the DNS registry. Instead, we propose hosts register themselves with the nameserver using a capability-based RPC system [12], which can then make intelligent load balancing decisions centrally.

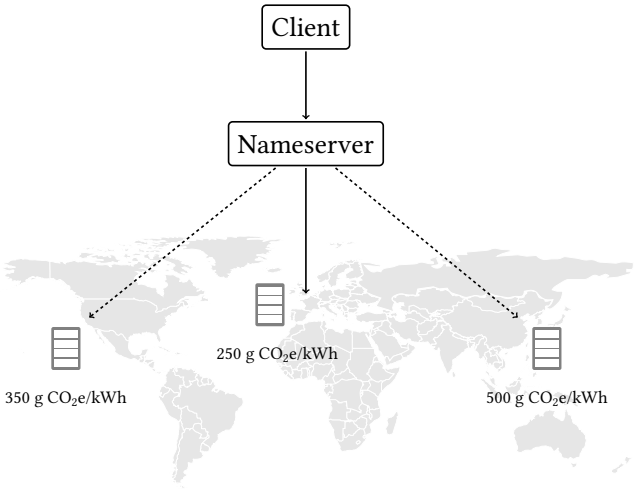

Figure 1: Carbon load balancing based on carbon intensity calculations.

Solar-based networks do not scale to the general state of the modern Internet. Estimating potential solar energy is insufficient for a truly carbon-aware name resolution given the diverse mix of sources that constitute a modern energy grid. An individual machine's carbon usage can be calculated from its power consumption and the electrical energy's carbon intensity. The carbon dioxide equivalent emissions of machine M (C_M) is calculated as the power used by machine M in kilowatt-hours (P_M) multiplied by the carbon intensity of the energy used by machine M measured in grams of carbon dioxide equivalent per kilowatt-hour (CI_M), $C_M = P_M \times CI_M$. A machine's power usage depends on its hardware components, their configuration and the machine's current work load. The carbon intensity of the energy grid varies due to decisions made by the grid operators and available energy mix which is possible to estimate [7, 9]. An increasing number of services and APIs are now available that can provide real-time carbon intensity information including ElectricityMaps [3] and `carbon-intensity.org.uk` [4]. There is also a power cost to transferring data, and a trade-off between distance to a server and the benefit in greener compute [8, 14].

All these parameters can form the inputs to a scheduling policy to minimise carbon emissions, that can use DNS load balancing as the mechanism to be implemented. The best policy will depend on the application's performance characteristics, and may even require predictive modelling of power sources for long-running stateful flows. Figure 1 shows a nameserver balancing load between geographically distributed servers, implementing a simple policy of returning the one with the lowest carbon intensity.

LOCO, 2024
2024.

3 WAKE-ON-DNS

The deployment of local networked services allows for low-latency, reliable, and offline operation. However, this decentralisation means the energy usage of a machine can't be amortized across many services in a mutli-tenant datacentre. Such local services are often running on idle on machines drawing non-trivial amounts of power for long periods of time [10]. This can be especially noticeable for non-public services like a local storage server, which can be idle for hours or days at a time depending on usage patterns. While power saving methodologies like low power CPUs, CPU sleep states, and Advanced Configuration and Power Interface (ACPI) can reduce this, they can only get us so far.

To address this, we extend carbon-aware name resolution to *actively* manage the power state of machines. We first allow machines to be put into an unpowered mode when idle for long periods of time. We can't do this in the DNS as there may be long-running flows that would be cut short. Instead, we leave this up to the server administrator or application software to configure, such as after an idle timeout period with no requests. We rely on a mechanism to wake the machine up on-demand, which can include: (1) cloud service provider APIs, (2) networked power supplies trigging a boot via a power cycle, (3) or the Wake-on-LAN (WoL) protocol. Before the machine goes to sleep we can register it's wakeup mechanism with the nameserver using an RPC interface.

Then, as a side effect of a name resolution for a service hosted on a machine, we wake the machine up. By specifying a low Time-To-Live (TTL) for the resource records associated with this name we can ensure that a name resolution will be the inception of any network request to the service.

Waking up the machines will take some amount of time, but if we delay responding to a DNS query until the server is ready we risk resolvers timing out and returning `SERVFAIL`. However, if a client sends a Transport Control Protocol (TCP) handshake to a machine that isn't up, yet it may time out, requiring retry logic on the client side. This is again a tradeoff between availability and latency and energy efficiency, as a powered-down machine can't immediately respond to a request but uses dramatically less power.

Aside from its use for a single machine, this active power management could be used to better implement load balancing across redundant servers. To reduce this to the simplest example, assuming a typical power curve two machines at a quarter load could be combined to one machine at half load, while the other is put to sleep. If the load exceeds the capacity of one machine, the other could be reanimated by the carbon-aware load balancer.

The WoL protocol is supported by a Network Interface Card (NIC) which listens for a particular packet format that contains the NIC's MAC address. Due to network configurations restricting IP broadcast this is typically only reachable on a local network. Proxy services for WoL exist, however, we can take advantage of DNS delegation to have the nameserver send WoL packets directly. Whenever a client queries the sleeping server, its query will be routed over the public Internet to the nameserver in the server's local network, which can then send a WoL packet to the server to wake it up, and respond as normal to the client's query, as illustrated by figure 2. This also works if the client queries from the local network.

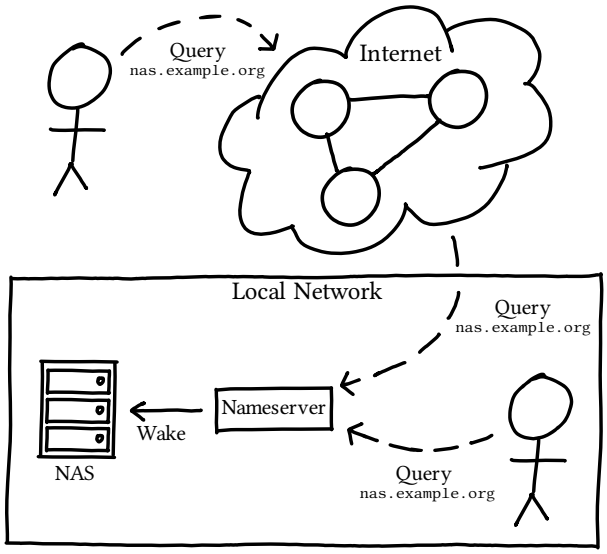

Figure 2: Local and global Wake-on-DNS.

4 OUR TALK

We will explain how the DNS can implement carbon-aware load balancing policies (2), how name resolution can actively manage the power state of machines (2), and will also explore how these mechanisms can enable existing applications to become carbon-aware.

Web-services. Much of the modern Internet is serving static content or responding to stateless requests over HTTP. As a result, such services are easily distributed as any server can be chosen to respond to the query. With a carbon-aware load balancing algorithm a webserver can be made carbon-aware with no changes to the client or application. And using active power management, we can actively scale the number of machines up based on load.

Mailservers. The DNS advertises mailservers with the use of `MX` records which return, for a queried domain, a set of mailserver domain names and their associated priorities [1, 2]. A mailserver domain can then be resolved to an IP address. Typically, the priory represents a backup server to use in the case of the higher priority servers failing. In order to make the Simple Mail Transfer Protocol (SMTP) carbon-aware the nameserver can modify these priorities to reflect the carbon intensity of the servers they map too, and put unused backup servers to sleep until they become required.

Local infrastructure. Many services are better suited to be situated on a local network instead of in a remote datacenter. However, this removes the economies of scale and centralised orchestration that typically comes with a datacenter environment. Consider, a research group's compute server that is periodically used for intensive batch processing, but for the majority of its uptime sits idle utilizing 100 Watts. Using DNS power management we can minimise the idle power usage of these machines by suspending them when they're not active, but keeping them available by waking them on a DNS request. And by doing this in the DNS, we interoperate with existing Internet protocols with no modification to the client or applications.

REFERENCES

[1] 1986. Mail routing and the domain system. RFC 974. https://doi.org/10.17487/RFC0974

[2] 1987. Domain names - implementation and specification. RFC 1035. https://doi.org/10.17487/RFC1035

[3] 2016. *Electricity Maps | Reduce Carbon Emissions with Actionable Electricity Data.* https://www.electricitymaps.com/

[4] 2017. *Carbon Intensity.* https://carbonintensity.org.uk/

[5] Tega Brain, Alex Nathanson, and Benedetta Piantella. 2022. Solar Protocol: Exploring Energy-Centered Design. In *Computing within Limits*. LIMITS. https://limits.pubpub.org/pub/solar/release/1

[6] Thomas P. Brisco. 1995. DNS Support for Load Balancing. RFC 1794. https://doi.org/10.17487/RFC1794

[7] Dr Alasdair R. W. Bruce, Lyndon Ruff, James Kelloway, Fraser MacMillan, and Prof Alex Rogers. 2021. Carbon Intensity Forecast Methodology. https://github.com/carbon-intensity/methodology/blob/9959aabaa17779f6c507d71c09386c864f8c07ff/Carbon%20Intensity%20Forecast%20Methodology.pdf

[8] Romain Jacob and Laurent Vanbever. 2023. The Internet of Tomorrow Must Sleep More and Grow Old. 3, 3 (2023), 27–32. https://doi.org/10.1145/3630614.3630620

[9] Diptyaroop Maji, Ramesh K. Sitaraman, and Prashant Shenoy. 2022. DACF: Day-Ahead Carbon Intensity Forecasting of Power Grids Using Machine Learning. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems* (New York, NY, USA, 2022-06-28) *(E-Energy '22)*. Association for Computing Machinery, 188–192. https://doi.org/10.1145/3538637.3538849

[10] David Meisner, Brian T Gold, and Thomas F Wenisch. 2009. PowerNap: Eliminating Server Idle Power. (2009). https://dl.acm.org/doi/pdf/10.1145/2528521.1508269

[11] Abel Souza, Shruti Jasoria, Basundhara Chakrabarty, Alexander Bridgwater, Axel Lundberg, Filip Skogh, Ahmed Ali-Eldin, David Irwin, and Prashant Shenoy. 2023. CASPER: Carbon-Aware Scheduling and Provisioning for Distributed Web Services. In *Proceedings of the 14th International Green and Sustainable Computing Conference*. 67–73. https://doi.org/10.1145/3634769.3634812 arXiv:2403.14792 [cs, math]

[12] Kenton Varda. 2014. *Cap'n Proto.* https://capnproto.org/

[13] Zhi Zhou, Fangming Liu, Yong Xu, Ruolan Zou, Hong Xu, John C.S. Lui, and Hai Jin. [n. d.]. Carbon-Aware Load Balancing for Geo-Distributed Cloud Services. In *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems* (2013). 232–241. https://doi.org/10.1109/MASCOTS.2013.31

[14] Noa Zilberman, Eve M. Schooler, Uri Cummings, Rajit Manohar, Dawn Nafus, Robert Soulé, and Rick Taylor. 2023. Toward Carbon-Aware Networking. 3, 3 (2023), 15–20. https://doi.org/10.1145/3630614.3630618