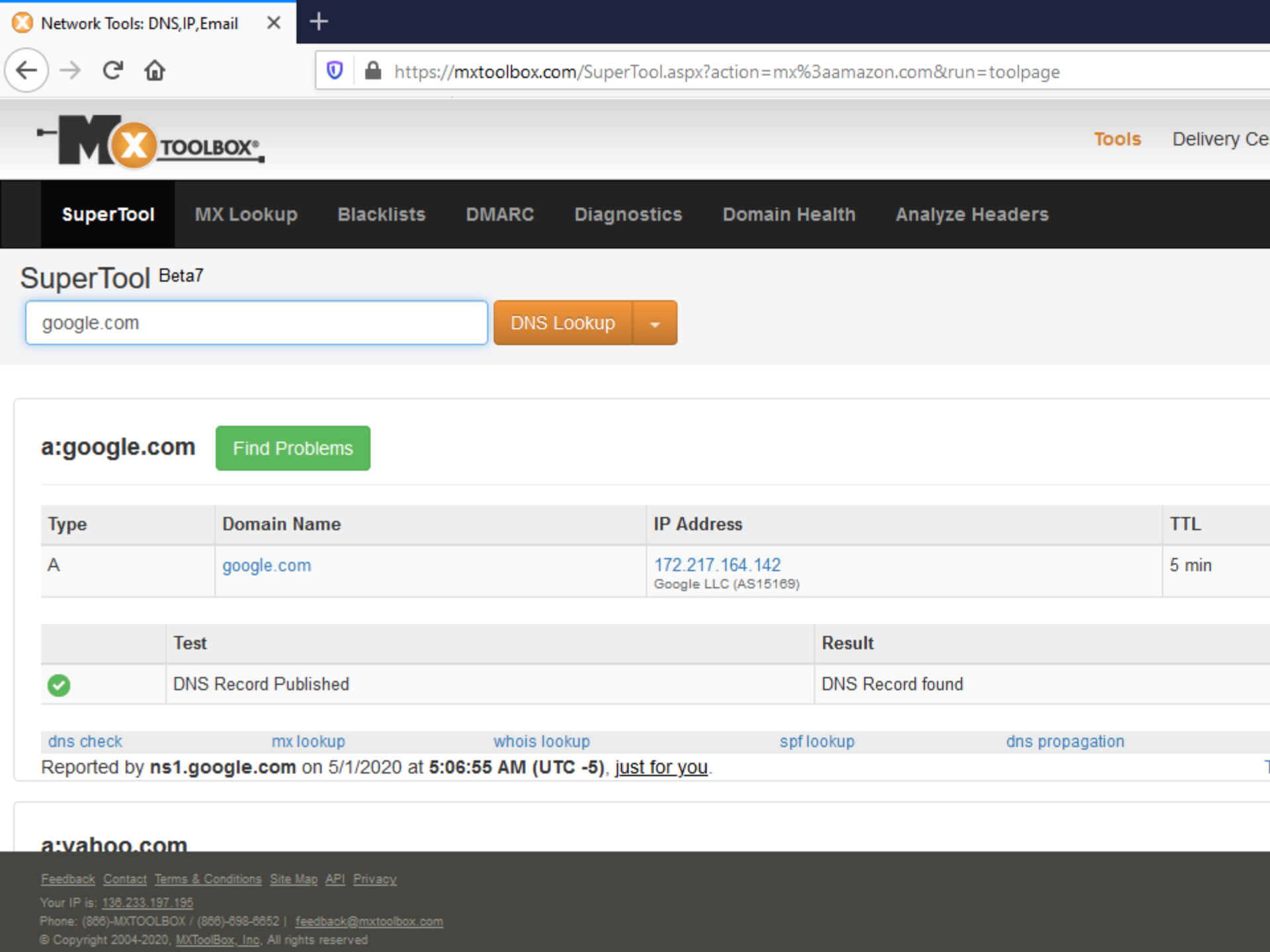
# Routing Requests to Data Centers

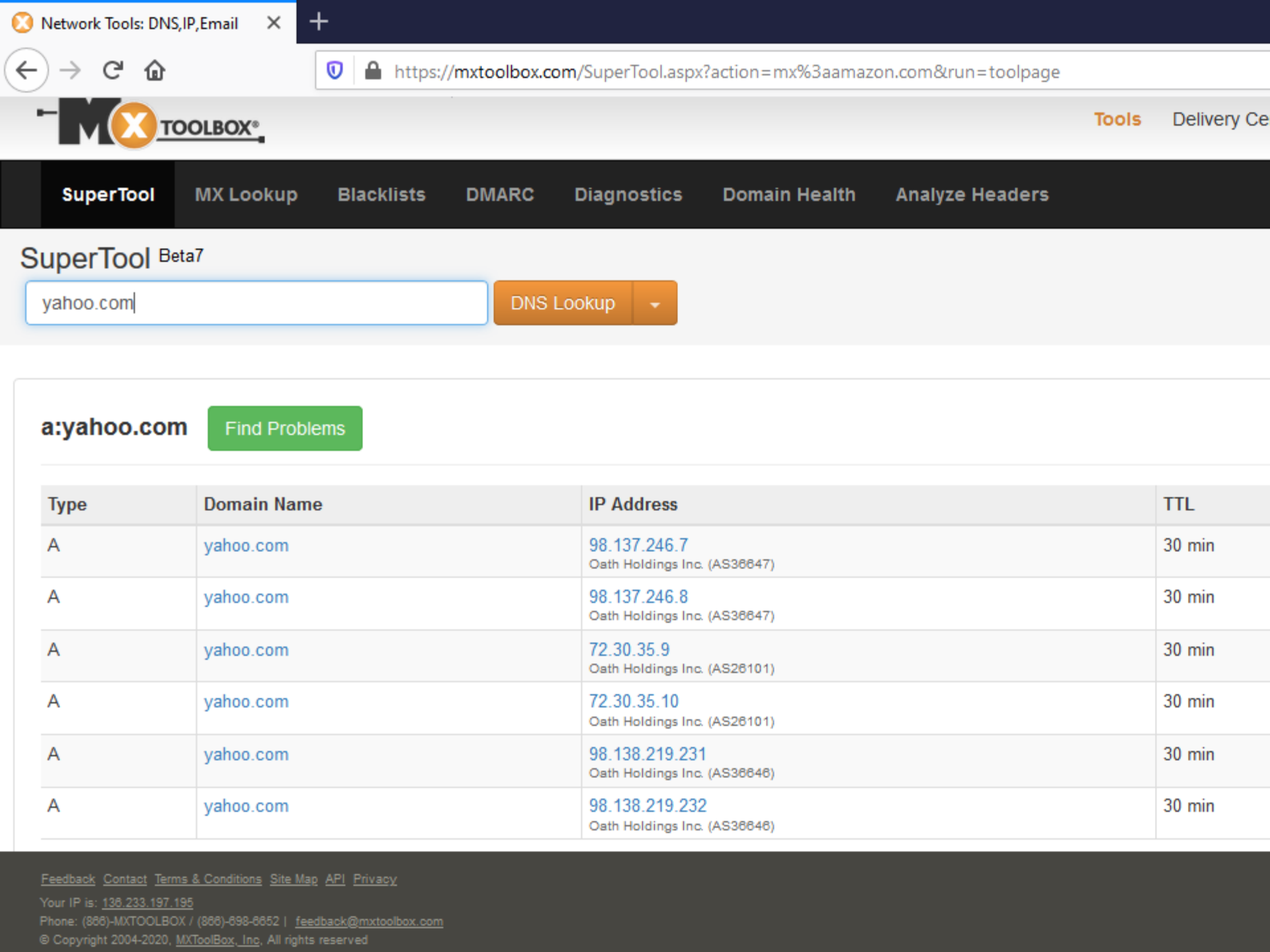
# Goal

- To understand how does your request for access (say to google.com, amazon.com etc.) gets to a data center over the internet
- Not really part of a "Data Center Network", which usually handles issues for networking within a single data center
- Still important to understand the overall end-to-end working at your level
- Assume you know basic Internet architecture and DNS
  - Interconnection between ISPs (direct peering or through Internet Exchange Points (IXPs))
  - Notion of Autonomous Systems and inter-AS routing like BGP
  - Basic DNS lookup process

# So what happens when you access some X.com?

- Since you started with a name, DNS must be involved first. So lets do a name lookup
- So DNS will look up X.com to get the IP of the server to connect to, right?
- Lets do some lookups, say google.com and yahoo.com


Network Tools: DNS,IP,Email
https://mxtoolbox.com/SuperTool.aspx?action=mx%3aamazon.com&run=toolpage
TOOLBOX
Tools
Delivery Ce
SuperTool
MX Lookup
Blacklists
DMARC
Diagnostics
Domain Health
Analyze Headers
SuperTool Beta7
google.com
DNS Lookup
a:google.com
Find Problems
Type
Domain Name
IP Address
TTL
A
google.com
172.217.164.142
Google LLC (AS15169)
5 min
Test
Result
DNS Record Published
DNS Record found
dns check
mx lookup
whois lookup
spf lookup
dns propagation
Reported by ns1.google.com on 5/1/2020 at 5:06:55 AM (UTC -5), just for you.
a:yahoo.com
Feedback
Contact
Terms & Conditions
Site Map
API
Privacy
Your IP is: 136.233.197.195
Phone: (866)-MXTOOLBOX / (866)-698-6652 | feedback@mxtoolbox.com
© Copyright 2004-2020, MXToolBox, Inc. All rights reserved


Network Tools: DNS,IP,Email
https://mxtoolbox.com/SuperTool.aspx?action=mx%3aamazon.com&run=toolpage
TOOLBOX
Tools
Delivery Ce
SuperTool
MX Lookup
Blacklists
DMARC
Diagnostics
Domain Health
Analyze Headers
SuperTool Beta7
yahoo.com
DNS Lookup
a:yahoo.com
Find Problems
Type
Domain Name
IP Address
TTL
A
yahoo.com
98.137.246.7
Oath Holdings Inc. (AS36647)
30 min
A
yahoo.com
98.137.246.8
Oath Holdings Inc. (AS36647)
30 min
A
yahoo.com
72.30.35.9
Oath Holdings Inc. (AS26101)
30 min
A
yahoo.com
72.30.35.10
Oath Holdings Inc. (AS26101)
30 min
A
yahoo.com
98.138.219.231
Oath Holdings Inc. (AS36646)
30 min
A
yahoo.com
98.138.219.232
Oath Holdings Inc. (AS36646)
30 min
Feedback
Contact
Terms & Conditions
Site Map
API
Privacy
Your IP is: 136.233.197.195
Phone: (866)-MXTOOLBOX / (866)-698-6652 | feedback@mxtoolbox.com
© Copyright 2004-2020, MXToolBox, Inc. All rights reserved

- google.com gives only one IP address, 172.216.164.142
- yahoo.com gives six IP addresses
- Does it mean google.com has only one server and yahoo.com has only six servers to handle all requests?
  - You know that cannot be true, however powerful the server(s) may be, given the volume of requests
- We will try to give you a high level description of what is happening
  - Can't go to all details given your background, but should still give you a good picture
- Disclaimer: google and yahoo were just used for name lookup illustration here. We are not trying to explain the exact process google/yahoo follows, but what are the broad technologies at play to make a request reach a data center in general

- Consider the scenario
  - A company operates multiple DCs geographically spread all over the world
  - Requests for services provided by the company come from all over the world
  - The company would like to direct these requests to the "appropriate" DC to give fast services
    - Load balancing across DCs
      - Do not send request to a highly loaded DC
    - Not send requests to a DC that may be temporarily down
    - Send requests to a DC that is close by to reduce network latencies
    - Any other policy or combinations thereof that the company may want

# Some Technologies at Play

- DNS Load Balancing
- Global Server Load Balancing (GSLB)
- Load Balancers (Hardware or Software)
- IP Anycast, Virtual IP and routing

# DNS Load Balancing

- A name may be associated with multiple physical machines
  - Multiple A records for the same name, or by the use of CNAME records in the zone file
- DNS server can give out all of them on query
- DNS client picks up one and returns it to the user application
  - The IP of the server that the application will connect to
- No standard definition of which one the client will pick, usually chooses the first one
- So DNS servers typically send the list in a cyclic order, so that each requests finally chooses a different server in round-robin fashion
  - Again, no standard specification. DNS server can send a subset also, including only one changing the order of the returned IP each time (basically not leaving the choice to the client)

- Pros
  - Simple scheme part of standard DNS implementations
  - Server sending in cyclic order and client choosing first one (commonly used) implements simple round robin load balancing.
- Cons
  - Round robin is known to be not good for load balancing when the jobs are not of same size
    - Think of 2 webservers, and 1st request is for a large webpage download, 2nd request is for a small webpage download, 3rd request is for a large webpage download, 4th request is for a small webpage download, and so on
  - Caching of name lookups can defeat the purpose
    - Will always give the same cached IP till cache expiry
- Not considered enough by itself for our purpose

# Global Server Load Balancing (GSLB)

- We have seen we must start with DNS, as all we have is a name (like google.com etc.)
- So it will go to a DNS server
- The problem was that the DNS server used a simple static policy
- GSLB – think of it as an "intelligent DNS"
- In addition to normal zone file lookup, GSLB server can
  - Keep track of which servers are actually up
  - Keep track of current load on the servers
  - Find the geographical origin of the request from source IP
  - Access any other relevant information available (example, internet traffic condition on specific routes etc.)

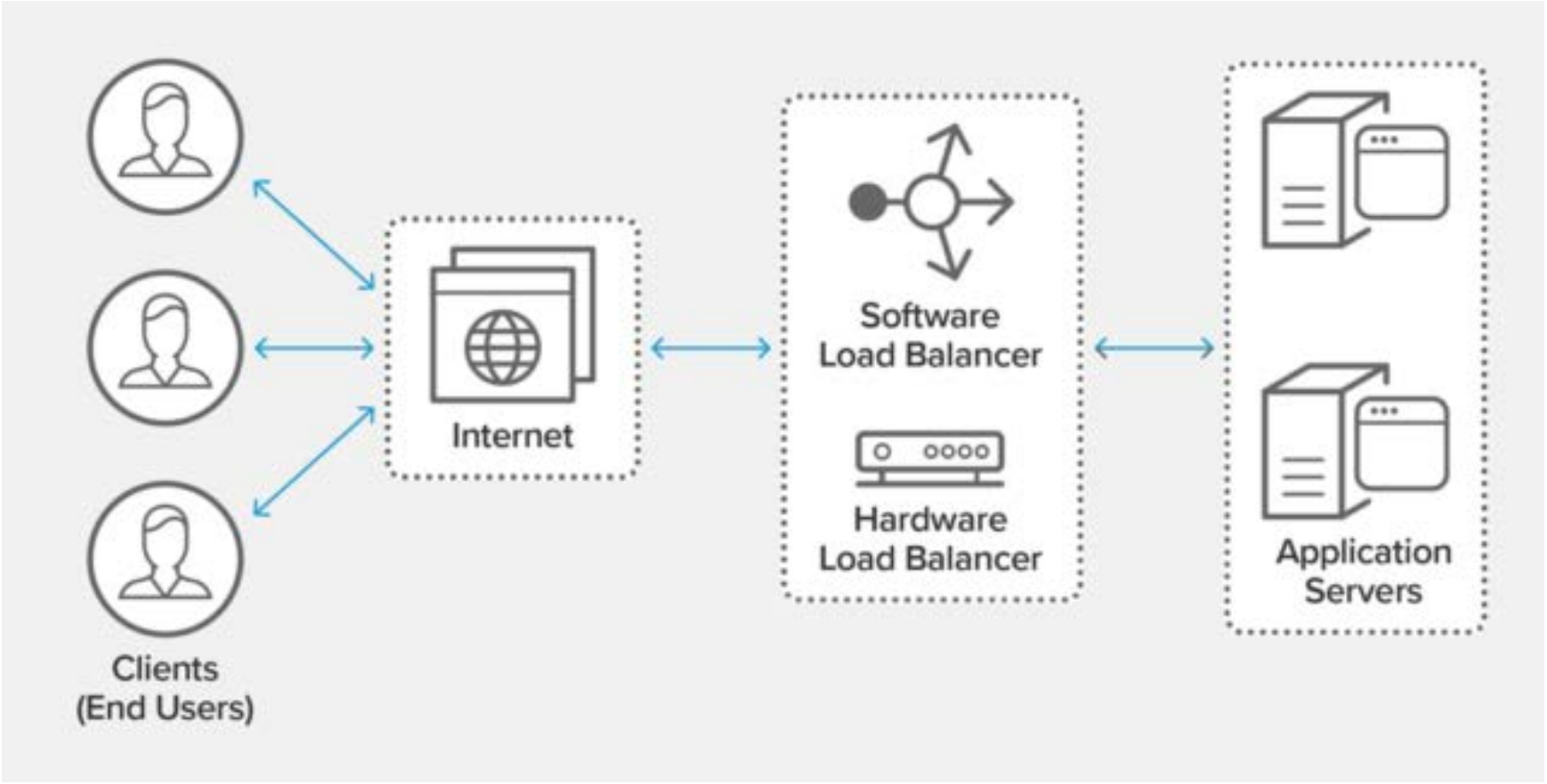
- Before sending the lookup result, GSLB server can apply any policy based on the dynamic information available and modify the final result sent to the client, based on any combination of
  - Do not send IPs of servers that are down
  - Send IPs of servers that are lightly loaded
  - Send IPs of servers located in a DC close to the origin of the client request
  - Any other change needed by the applied policy
- The caching problem of normal DNS still remains

- Implementing a GSLB server
  - Can integrate with DNS code
    - Not modular or flexible
  - Implement as a proxy to the DNS server
    - GSLB server IP given as DNS nameserver address
    - Traps all DNS requests
    - Queries the normal DNS server to get the lookup results
    - Modifies the result as per policy applied and sends to client
  - Can host in cloud, with name lookups in DNS server redirecting to the server in the cloud
    - Many companies provide cloud based load balancing (for example, CloudFlare)

- Is GSLB enough for our scenario?
  - A large DC will have tens to hundreds of thousands servers
  - Then you can have multiple DCs for a company
  - It is not possible for GSLB servers to keep track of all servers in all DCs dynamically and send out the IP of a single server (or a small subset of servers from this very large number) to the client
  - Even the DNS zone file cannot have that many entries against a name
    - Huge text file, lookup will be too slow
    - Returned result will be too large
    - Needs change in zone file every time a new server is added/deleted or its IP changed in some DC changes IP etc.
  - Simply not scalable by itself

# Load Balancers

- Specialized devices that balances incoming load among a set of servers
- Monitors the status and load of the servers continuously
- Directs an incoming request/connection to a server based on current load condition
- Can be implemented as
  - Specialized hardware device (also called Application Delivery Controllers in many cases)
    - Some major OEMs – F5, Citrix, Radware, Fortinet, Barracuda, …
  - Virtual Appliances (pre-configured VM image)
    - All hardware LB/ADC vendors also give this option
  - Software load balancers (software that can be installed on commodity servers)
    - Example – nginx, HAproxy, Neutrino….
  - Cloud based (invoked as a service run from the cloud)
    - Example – Cloudflare




courtesy: www.nginx.com

# A Feasible Architecture for Request Routing

- Each DC has a small set of $N$ load balancers
- All servers in the DC are partitioned into $N$ clusters, with all servers in each cluster connected to one load balancer
- Each load balancer monitors and collects information of the $N$ servers
- Each load balancer makes a summary load information, which is fed to the GSLB server
- The IPs against the service name (say www.X.com) in the DNS server are the IPs of these load balancers
- GSLB server monitors these load balancers

- Client sends request for name lookup for www.X.com
- GSLB server traps the request, sends to DNS server for name lookup
- DNS server gives IPs of load balancers of DCs
- GSLB server applies policy and returns IP of one (or a small subset) of load balancers to the client
- Client sends its service request to the load balancer
- Load balancer directs the request to an appropriate server connected to it
- Server provides the service and sends result to the load balancer
- Load balancer sends the result back to client
- Lots of details underneath
  - For example, try thinking of actual TCP connections to be made to access www.X.com. Good exercise to at least see the problems

- Can extend to multiple levels of load balancers inside a DC
  - Top level load balancers connect to second level load balancers and balance requests to them
  - Second level load balancers connect to third level load balancers and balance requested to them
  - And so on…
  - Each lowest level load balancer  connects to servers in a cluster
  - No. of levels depend on many factors, including number of servers, load balancer capacity, expected request rate, latency constraint etc.

- For a company operating multiple DCs each of which can handle the request, usually the GSLB server will choose the DC based on policy
  - Geographically closest to request source based on source IP to reduce latency (for example, for streaming content delivery)
  - Avoid DCs that may be down (for example, due to failures or scheduled maintenance)
  - Send to DC with lower load
  - Any other policy based on any other information available (for example, internet traffic along some routes, route outages etc.)
  - Any policy based on combinations of the above
- The GSLB server may be hosted outside company DCs at managed DNS providers, or can be inside DCs also
- The sever load balancers will be inside DCs
- Note that GSLB is also load balancing, and sometimes goes by other names too in practice
  - Terminologies can be confusing sometimes

# IP Anycast

- Another very useful way to route client requests to different DCs (usually topologically closest DC)
- Multiple machines (maybe in different parts of the world) can share the same IP
- Configuration in BGP routers
- Multiple BGP routers advertise the routes with the same IP
- When a user request for the anycast IP comes, the BGP router selects the route with the shortest distance
  - Routes to "nearest" server with that IP
- Used extensively for routing requests to nearest DC/Server providing service being sought
- Can also use some other parameters for better load balancing

# Virtual IP

- An IP that is not assigned to any physical network interface, but is backed by at least one physical network interface (usually more than one)
- The same virtual IP can be assigned to one or more network interfaces, on the same machine or different machines
- Gives a one-to-many mapping of one virtual IP to multiple physical IPs
- Only the virtual IP is advertised outside
  - The user sees only one IP for the service/server
- When a packet comes for the virtual IP, it is assigned to one of the physical IPs by an end device (for example, routers)
- The same virtual IP can be assigned to a new device (for example, a new server in a cluster)
  - External world is completely unaware
- Used for many applications including NAT'ing, load balancing, failover etc.
  - Full discussion of virtual IP beyond our scope here

- An example use in our scenario
  - Suppose that in a DC, there are two top level load balancers, one active (working) and one passive (standby, to be used if first one fails). Each has a physical IP, say 10.1.1.5 and 10.1.1.6
    - At least 2 load balancers needed for fault tolerance
  - Assign the same virtual IP, say 200.100.1.5 to both
    - Internal routing algorithm gives two routing entries in the router mapping 200.100.1.5 to both 10.1.1.5 and 10.1.1.6. Say 10.1.1.5 advertises itself with higher priority than 10.1.1.6.
  - DNS lists only 200.100.1.5 against the name for accessing the service (say www.X.com)
  - Router advertises 200.100.1.5 to the external world
  - User request sent to 200.1000.1.5 comes to the router following standard internet routing

- Router sees two entries, one with higher priority, sends to 10.1.1.5
  - So normally nothing goes to 10.1.1.6
- If 10.1.1.5 fails, it is detected by local routing algorithm, and the entry is removed from the router
- Subsequent requests to 200.100.1.5 goes to 10.1.1.6 as it is the only entry in the router
- If 10.1.15 comes up again, it is again discovered by the routing protocol (with higher priority), entry is added to router, and used as the primary server for subsequent requests
- Can be extended to any number of top-level load balancers
  - Assign the same virtual IP to them, just more entries in router
- Can be used in active-active mode also (all load balancers take requests at all times)
  - Assign same routing priority. Router will send in round-robin fashion usually

- Advantages
  - No DNS change needed when load balancers are added or removed
  - No DNS caching problem, as the same virtual IP is always valid
  - Only local routing table change after server addition/failure/removal, usually very fast
- Just gave you an overview, lots of actual details skipped that are beyond the scope here
  - Can always try to learn on your own later, the keywords and basic processes are all here ☺