ANYCAST AS A LOAD BALANCING FEATURE

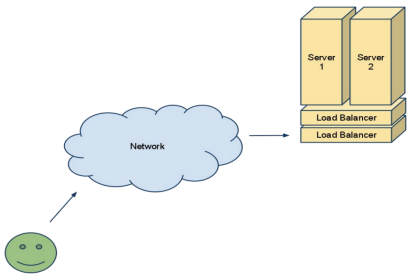
(IWS TERM PAPER)

ABSTRACT

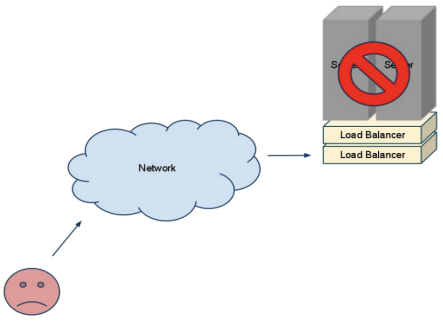
Anycast is a networking and routing technique that allows multiple servers or network devices to share the same IP address. When a user or device sends a request to that IP address, the network routes the request to the nearest or most optimal server in the Anycast group based on various factors such as network proximity, load, or defined policies. This helps distribute the load and improve the efficiency and reliability of services, making it a valuable tool for enhancing the availability and performance of websites and network services. Google's IT organization consists of numerous sub-teams, each responsible for specific services such as DNS, LDAP, HTTP proxy, and more. These sub-teams have global deployments with their replication methods. Another team provides Load Balancing and failover services, which other teams within Google can utilize without needing to manage the underlying technology. Transitioning the Anycast routing configuration to a managed load balancer service streamlines the service setup process, making it less work and simplifying the complexities involved. This move also grants the advantage of swift and automatic failover, taking into account proximity to ensure efficient service continuity. Furthermore, it contributes to a reduction in the load and complexity of the network infrastructure by consolidating service advertisements into a single peering point per site. Additionally, this approach decreases the frequency of routing changes, limiting them to situations where entire sites experience failures. This paper elaborates on the workings of Anycast, its advantages, and the architectural approach used to provide it as a failover service within Google.

INTRODUCTION

The concept of Load Balancing is most intuitively understood by deploying the necessary service replicas in a server room and employing a Load Balancer to distribute the workload among them.



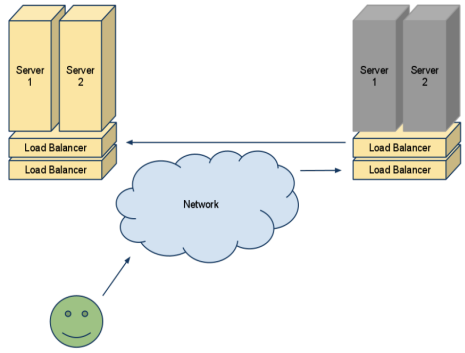
Drawing 1: Regular Load Balancing Scenario



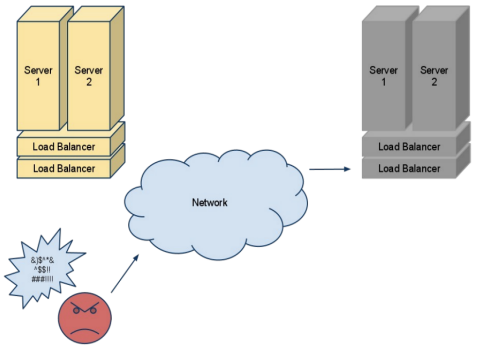
Drawing 2: Failure mode

To enhance reliability, Load Balancers are commonly deployed in pairs for High Availability, an assumption they maintain in this paper. A typical Load Balancing scenario(refer to Drawing 1). While this setup already improves reliability, further considerations are possible. Envision a scenario where users and the Load Balancer are active, but all backends for a particular service are inactive in the event of a disaster. The current solution, on its own, would not address this problem(refer to Drawing 2). An improved design would seamlessly reroute all those clients to an alternate location or server room, ensuring the process is as transparent as possible. One method to achieve this is by identifying the nearest secondary location and configuring the Load Balancer to proxy or redirect all user requests there until the local service is restored. Many load balancing products include automatic redirection features, illustrated on the right (refer to Drawing 3). However, in the event that the Load Balancers are also unavailable (refer to Drawing 4), ensuring users can access an alternative instance of the service becomes crucial. How can this be achieved with minimal disruption? Various solutions exist for this problem, contingent on the specifics of each implementation. One potential approach involves updating DNS

records for the services, enabling users to reach the service in a different location. Possibly, automating this DNS update is feasible, but it requires a mechanism to monitor the status of services in other locations and keep track of their condition. This ensures that the system can redirect users appropriately in the event of a failure. Given that services are often deployed across numerous locations, centralizing all information about services may not be efficient. Therefore, the DNS update mechanism should be distributed to every location where the service is deployed. However, this is not considered an optimal solution, as integrating monitoring and automatic failover into the existing Load Balancing infrastructure is a more favorable option.



Drawing 3: Remote failover



Drawing 4: Remote failover - failure mode

The use of DNS TTL (Time to Live) can pose challenges as well. In some cases, employing very small TTLs may not be possible, and the time it takes for DNS changes to propagate still results in downtime from the users' perspective. Once the system is restored, there is also the necessity to update DNS entries again to direct users back to the original location.

GOOGLE'S IMPLEMENTATION

Google integrates Anycast for failover between Load Balancing clusters, extending the benefits of Anycast to all services behind Google's Load Balancers. This simplifies the network environment significantly by minimizing the number of machines advertising routes. Google's approach utilizes BGP, establishing a hierarchy for route advertising, although alternative protocols are also viable. Anycast eliminates the need for remote failover using proxies, providing a cleaner solution where clients directly connect to the failover location. This method ensures the retention of client identifying information and eradicates the proxy overhead between servers and users, illustrated in Drawing 3 (Remote failover) and Drawing 4 (Remote failover - failure mode). Google's dedicated team offers Load Balancing as a service, ensuring complete independence from the specific service setup. Multiple services can leverage the same Load Balancing infrastructure, and scaling the number of service replicas doesn't complicate the network design due to a controlled number of route advertisers.

Another benefit of having Load Balancers as Anycast peers is the reduced number of routing changes. The Load Balancer consolidates multiple service instances into one VIP, addressing concerns related to Anycast deployments. The Load Balancer oversees service-specific state health checks, facilitating Anycast deployment for both UDP-based and TCP-based services. Google has configured the network environment with a reserved subnet for all Anycast virtual IPs (VIPs). Routers are configured to accept /32 route advertisements in that subnet from the Load Balancers, implementing protection against misconfigurations through ACLs that exclusively allow routes from the specified subnet. This prevents accidental takeover of IP space, and Anycast VIPs can be configured alongside normal VIPs on the same Load Balancers.

SOFTWARE USED BY GOOGLE

All Google's Load Balancers are deployed in high availability pairs to safeguard against single machine failures. For this purpose, Google utilizes Heartbeat from the Linux-HA project[2] as a cluster resource management software. Heartbeat manages the up and down states of network interfaces and backend management software, configured as heartbeat resources. For backend monitoring and failover, Google employs ldirectord[4]. Ldirectord conducts health checks against the backends for each VIP, dynamically adjusting the Load Balancing pool by adding or removing service instances based on their health state. In the event of a failure in all backends, ldirectord has a fallback option to redirect all connections to a different location.

Google introduced a feature to ldirectord, implementing a fallback command configuration. When the last of the local backends goes down, it triggers this command, utilized to bring the Anycast IP address up or down based on the backend status. Ldirectord communicates directly with ifconfig (for IP management) and ip_vs (via ipvsadm) to manipulate service backends in the Load Balancing pool. The Linux Kernel module for Load Balancing, ip_vs[1], supports various modes, including tunneling, half NAT, and Direct Routing (DR). In Google's setup, all VIPs are configured using DR. Quagga[3], a network routing software package, is employed to enable Google's Load Balancers to advertise BGP routes to network devices. For each service, Google allocates an IP and configures standard LVS settings to manage VIPs. The added feature in ldirectord facilitates bringing the Anycast IP network interface up or down based on backend status. If all backends are down, ldirectord brings the IP from that VIP down, and Quagga promptly notifies routers to withdraw the route to that VIP.

ADDING NEW SERVICES TO THE SETUP

Incorporating services into Anycast becomes a straightforward process as they configure the backends within a VIP on a Load Balancer enabled for Anycast. In Google's context, this involves adjusting Heartbeat and ldirectord settings. The pre-established network configuration significantly reduces the entry barrier for new services. The expansion of a service to additional locations follows an identical process, with Anycast routing seamlessly directing user traffic to the new, closer load balancer setup.

FAILURE MODES AND RECOVERY TIMES

The mentioned recovery times take into consideration Google's specific Anycast deployment. Environments with different timeouts and configuration parameters may experience varying response and recovery times. The route propagation time is less than 1 second, and Google has set a 30-second "dead timer" for the routers to consider the BGP peer dead. A clean stoppage of the BGP peering service causes an outage of less than 1 second for Google's services as the routes update. In the scenario where all service backends become unavailable, it will take the service-specific health check time plus the less than 1-second route propagation delay for recovery.

In the event of a sudden network or power failure at the location, it will take the time for the "dead timer" to expire plus the small route propagation delay for recovery.

DISCUSSION

In exploring the above context, Google's decision to transfer Anycast routing configuration to a managed load balancer service emerges as a strategic move for streamlining operational tasks and simplifying service setup. By doing so, Google ensures its services benefit from a rapid and automatic failover system that takes distance into account. This transition not only reduces the workload and complexity associated with service configuration but also has a positive impact on the overall efficiency of the network infrastructure. Google's approach of aggregating service advertisements into a single peering point per site contributes to a more streamlined and cohesive system. Moreover, the deliberate reduction in the frequency of routing changes, limited to complete site failures, reflects a thoughtful strategy aimed at enhancing the reliability and stability of the network.

ANALYSIS

Looking closely at how Google is changing its Anycast routing setup, it seems like a smart move. By using a managed load balancer service, they make it easier to set up and manage their services. This means things run more smoothly. Google is also making sure their services can quickly and automatically switch to a backup location if needed, and they're considering the distance for better performance. The choice to group service advertisements into one peering point per site simplifies how their network is set up and used. They're intentionally making fewer changes to the way data travels, mainly when there's a big issue at a site. This shows Google is thinking carefully about how to keep their network running well. Overall, it looks like a smart and efficient move by Google.

CONCLUSION

Shifting Anycast routing configuration to a managed load balancer service simplifies the tasks for Google and makes service setup easier. This change ensures Google's services benefit from quick, automatic, distance-aware failover. It also cuts down on the workload and complexity of the network system by consolidating service advertisements into a single peering point per site. Additionally, it lowers the frequency of routing changes, limiting them to complete site failures.

REFERENCES

[0] Anycast as a Load Balancing feature - Anycast
[1] The Linux Virtual Server Project, http://www.linuxvirtualserver.org
[2] High Availability, http://www.linux-ha.org
[3] Quagga, a software routing suite, http://www.quagga.net
[4] Ldirectord, http://www.vergenet.net/linux/ldirectord/