International Journal of Multidisciplinary Research and Growth Evaluation.

AWS Network Load Balancer (NLB) for High-Performance Traffic Routing: A Research-Based Study

Satish Yerram
Independent Researcher, USA

* Corresponding Author: **Satish Yerram**

Article Info

ISSN (online): 2582-7138
Volume: 05
Issue: 02
March-April 2024
Received: 08-03-2024
Accepted: 09-04-2024
Page No: 1139-1141

Abstract
Modern cloud-based applications demand high-performance, fault-tolerant, and resilient networking architectures that can support millions of requests per second while maintaining extremely low latency. To meet these requirements, Amazon Web Services (AWS) provides the Network Load Balancer (NLB) as part of its Elastic Load Balancing (ELB) family. NLB operates at Layer 4 of the OSI model, focusing on TCP, UDP, and TLS traffic routing with deterministic performance. Unlike Application Load Balancer (ALB), which supports path- and host-based routing at Layer 7, NLB is designed to handle vast volumes of transport-level traffic by using connection-based hashing algorithms. This paper provides a research-driven overview of NLB's architecture, routing logic, target group configurations, deployment patterns, benefits, challenges, and best practices. By examining the operational principles and architectural trade-offs of NLB, this study highlights its importance for latency-sensitive workloads, hybrid cloud deployments, and highly scalable microservices environments.

DOI: https://doi.org/10.54660/.IJMRGE.2024.5.2.1139-1141

Keywords: AWS NLB, NLB, Network Load Balancer

1. Introduction

Load balancing is a fundamental component in achieving scalability, fault tolerance, and resilience in distributed systems. AWS offers three primary types of load balancers—Classic Load Balancer (CLB), Application Load Balancer (ALB), and Network Load Balancer (NLB). Among these, NLB is specifically designed for scenarios where ultra-low latency and deterministic traffic flow are required. Unlike ALB, which provides sophisticated Layer 7 content-based routing such as path or host header matching, NLB is a Layer 4 load balancer that routes requests purely on the basis of network and transport-level information. This makes it a strong candidate for latency-sensitive use cases such as financial trading systems, real-time communications platforms, gaming applications, and high-throughput APIs. By combining static IP addressing, TLS passthrough, and zonal isolation, NLB provides a balance between performance, scalability, and security.

2. Architecture of NLB in AWS

The architecture of NLB is designed around high scalability and availability. Each NLB is provisioned across multiple Availability Zones (AZs) in a Virtual Private Cloud (VPC) and can be assigned static Elastic IP addresses, one per AZ, ensuring predictable connectivity for clients. When traffic arrives at the NLB, it is routed using a deterministic flow hashing algorithm that considers the five-tuple of protocol, source IP, source port, destination IP, and destination port. This ensures that packets belonging to the same connection are always routed to the same backend target, maintaining consistency for stateful applications. NLB also supports zonal isolation, meaning that each Availability Zone operates independently; in case of failure in one AZ, other zones continue to serve traffic without disruption. Cross-zone load balancing can be enabled to distribute requests evenly across all registered targets, but administrators must weigh this against potential inter-AZ data transfer costs. NLB also supports TLS termination, allowing it to either forward encrypted traffic (TLS passthrough) or decrypt traffic at the load balancer level

and re-encrypt it before forwarding to targets.

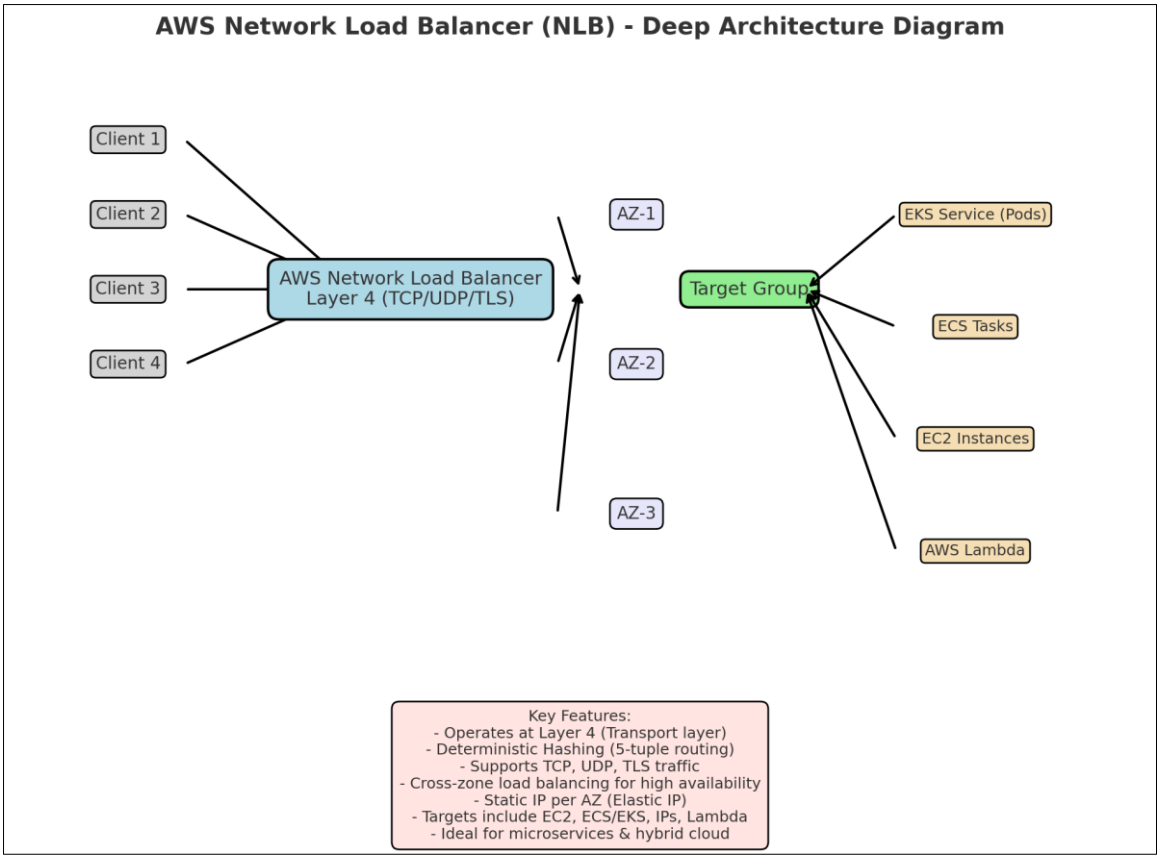

Fig 1: AWS Network Load Balancer (NLB) architecture with clients, Availability Zones, target groups, and microservices.

3. Traffic Routing and Distribution Logic

Unlike ALB, which can inspect application-level headers and routes traffic based on path or host information, NLB focuses on transport-level traffic management. Its routing logic is based on connection-level load distribution, where flows are consistently mapped to targets using deterministic hashing. Once a flow is assigned to a target, all packets within that session continue to be routed to the same backend until the connection terminates. This ensures low latency and reduces jitter, which is essential for applications such as voice over IP (VoIP), streaming services, and gaming platforms. NLB also provides flexible health checks, supporting both TCP and HTTP(S) checks to ensure only healthy targets receive traffic. These health checks can be tuned for intervals, thresholds, and timeouts to meet workload-specific needs. Additionally, administrators can configure idle connection timeouts to control how long inactive connections are maintained before being reset.

4. Target Groups and Supported Targets

NLB directs traffic to registered targets through target groups. These target groups can consist of Amazon EC2 instances within the same VPC, containers running on Amazon Elastic Kubernetes Service (EKS) or Elastic Container Service (ECS), private IP addresses from on-premises systems in hybrid cloud setups, and AWS Lambda functions. By supporting both IP-based and instance-based targets, NLB enables a flexible architecture that extends load balancing beyond the boundaries of AWS into hybrid environments. Each target group is associated with its own set of health checks, ensuring granular control over routing behavior. For containerized workloads, NLB can dynamically register and deregister IP addresses as pods scale up and down, making it particularly suitable for Kubernetes-based microservices environments. The ability to route traffic to AWS Lambda functions also adds serverless integration, where event-driven workloads can process traffic without dedicated infrastructure.

5. Deployment Scenarios and Use Cases

NLB is widely adopted in scenarios that require high throughput and consistent low latency. In the financial services sector, it is used for trading applications where microseconds matter, while in telecommunications it supports VoIP and messaging workloads. For gaming platforms, NLB ensures smooth player experiences by maintaining stable connections with low jitter. In hybrid cloud deployments, organizations can register on-premises IP addresses as NLB targets, enabling seamless routing between AWS and local data centers. Another common deployment pattern involves combining NLB with ALB: NLB provides static IP addresses and TLS termination at the edge, while ALB handles advanced Layer 7 routing such as host- and path-based rules. This hybrid design balances performance with flexibility, allowing enterprises to serve a diverse range of workloads securely and efficiently.

6. Benefits and Limitations

NLB offers several advantages that distinguish it from other load balancers in AWS. Its primary benefits include ultra-low latency, the ability to scale to millions of connections per second, and support for static IP addresses, which are critical

for security whitelisting and integration with external firewalls. NLB's native integration with ECS, EKS, and Lambda makes it suitable for modern application architectures, while its zonal isolation ensures high availability and resilience against single-AZ failures. However, NLB also comes with limitations. It lacks Layer 7 intelligence, meaning it cannot inspect or act on HTTP headers, cookies, or query strings. Observability is also more limited compared to ALB, with fewer metrics and no direct access to application-level logs. Furthermore, cross-zone load balancing can result in additional costs, and debugging network flows often requires VPC Flow Logs and CloudWatch metrics.

7. Challenges and Best Practices

Operating NLB effectively requires careful attention to monitoring, cost management, and hybrid configurations. One challenge is limited visibility into application-level traffic, which makes it important to integrate NLB with CloudWatch metrics and VPC Flow Logs for detailed flow analysis. Cost optimization is another concern, as enabling cross-zone load balancing may increase inter-AZ data transfer expenses. To address these issues, best practices include designing target groups with proper scaling policies, using TLS certificates from AWS Certificate Manager (ACM) for secure traffic handling, and combining NLB with ALB or API Gateway when Layer 7 functionality is required. Additionally, workloads that demand fine-grained security should employ IAM policies, security groups, and network ACLs in combination with NLB's transport-level routing to maintain compliance and control.

8. Backend Communication with Microservices through NLB

One of the most significant advantages of using AWS Network Load Balancer is its ability to enhance backend communication in microservices-based architectures. Modern applications deployed on Amazon Elastic Kubernetes Service (EKS) or Amazon Elastic Container Service (ECS) often consist of hundreds or thousands of microservices that must communicate reliably and efficiently. NLB, operating at Layer 4, is optimized for speed and throughput, allowing it to maintain millions of concurrent connections with minimal latency. By leveraging deterministic hashing algorithms, NLB ensures that network flows are consistently routed to the same microservice backend, which is particularly beneficial for stateful services where connection persistence is important. Another key benefit of NLB in microservices environments is its ability to handle sudden traffic spikes. Because it is designed for high scalability, NLB can absorb large bursts of requests without performance degradation, ensuring seamless service-to-service communication. In Kubernetes environments, NLB integrates directly with Service objects, allowing pods to be registered and deregistered dynamically as they scale in and out. This results in an infrastructure where communication between services remains stable even as workloads change. Additionally, NLB supports both TCP and UDP traffic, making it suitable for a wide range of inter-service communication protocols, including gRPC, HTTP/2, and custom binary protocols often used in high-performance systems. This protocol flexibility ensures that developers are not constrained by application-layer routing limitations and can build microservices that maximize throughput. From a performance perspective, NLB minimizes overhead by avoiding packet inspection at higher layers, enabling near-native network speeds when routing traffic between services. This combination of scalability, speed, and deterministic routing makes NLB particularly well-suited for microservices architectures where backend communication efficiency directly impacts overall system performance.

9. Conclusion

AWS Network Load Balancer plays a critical role in cloud-native architectures where performance, scalability, and deterministic routing are paramount. While it does not provide context-based routing like ALB, its ability to handle millions of connections with ultra-low latency makes it ideal for performance-sensitive workloads. With support for multiple target types including EC2, containers, IP addresses, and Lambda, NLB enables both cloud-native and hybrid cloud deployments. Enterprises often use it alongside ALB to combine transport-level performance with application-level intelligence. By following best practices in monitoring, cost control, and security, organizations can leverage NLB to build resilient, scalable, and high-performance architectures in AWS.

10. References

1. Amazon Web Services. Elastic Load Balancing: Network Load Balancer Features. AWS Documentation; 2021. Available from: https://docs.aws.amazon.com/elasticloadbalancing
2. Amazon Web Services. How Network Load Balancer Works. AWS Documentation; 2020. Available from: https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html
3. Amazon Web Services. Target Groups for Your Network Load Balancer. AWS Documentation; 2022. Available from: https://docs.aws.amazon.com/elasticloadbalancing/latest/network/load-balancer-target-groups.html
4. AWS Containers Blog. Integrating Amazon ECS and EKS with NLB. 2021. Available from: https://aws.amazon.com/blogs/containers
5. AWS Architecture Center. Patterns for Using ALB and NLB Together. 2022. Available from: https://aws.amazon.com/architecture
6. Amazon Web Services. Monitoring Network Load Balancers. AWS CloudWatch Documentation; 2022. Available from: https://docs.aws.amazon.com/elasticloadbalancing/latest/network/load-balancer-cloudwatch-metrics.html
7. Amazon Web Services. Securing Load Balancers in AWS. AWS Security Best Practices; 2020. Available from: https://docs.aws.amazon.com/security