Owf Flow

Project Requirements Document

Overview

Owl Flow is a modern load balancer designed to efficiently distribute network traffic across multiple servers, improving performance, reliability, and scalability.

Key Features of Owl Flow

1. **Resource Optimization** – Evenly distributes workload among servers to prevent overload.
2. **High Availability** – Automatically reroutes traffic if a server fails.
3. **Improved Performance** – Reduces response time by intelligently distributing requests.
4. **Scalability** – Easily add new servers to the pool without downtime.

Supported Load Balancing Types

1. **Layer 7 (Application Layer)** – HTTP/HTTPS-based balancing, ideal for web applications.
2. **GSLB (Global Server Load Balancing)** – Geographic traffic distribution for global networks.

Load Balancing Algorithms

Owl Flow supports multiple distribution strategies:

1. **Round Robin** – Requests are cycled between servers in sequence.
2. **Least Connections** – Traffic is sent to the server with the fewest active connections.
3. **IP Hash** – A client is consistently directed to the same server (useful for sessions).
4. **Weighted Round Robin** – Servers handle traffic according to their capacity (weights).
5. **Least Response Time** – Requests are routed to the fastest-responding server.

Functional Requirements

Web Interface

Authentication

As an Admin, I want to sign in to the web interface using:

- Email
- Password

so that I can access the load balancer dashboard.

As an Admin, I want to receive an error message when entering incorrect credentials.

As an Admin, I want to sign out of the web interface to secure my session.

Load Balancer Management

As an Admin, I want to:

- Create a new load balancer (LB) instance
- Specify its name (e.g., “Production-LB”)
- Select a default algorithm (Round Robin/Least Connections/etc.)

so that I can start distributing traffic.

As an Admin, I want to delete an existing LB when it’s no longer needed.

Server Management

As an Admin, I want to:

- Add a backend server to an LB by entering:
  - IP address/hostname
  - Port (e.g., 80, 443)
  - Optional weight (for Weighted Round Robin) so that traffic can be routed to it.

As an Admin, I want to remove a server from the LB pool during maintenance.

As an Admin, I want to temporarily disable a server (without deleting it) for troubleshooting.

Monitoring & Metrics

As an Admin, I want to see real-time statistics for each LB, including:

- Total requests per second (RPS)
- Average response time (in ms)
- Error rates (4xx/5xx) per server

so that I can assess performance.

As an Admin, I want to view a live list of all servers with their:

- Current status (Online/Offline/Overloaded)
- CPU/RAM usage (if metrics are available)
- Active connections count

so that I can identify bottlenecks.

As an Admin, I want to configure health check intervals (e.g., every 1-5m) and thresholds (e.g., 3 critical errors = offline).

Alerts & Notifications
As an Admin, I want to receive browser notifications (or emails) when:

- A server fails health checks.
- Average latency exceeds a threshold (e.g., 500ms).
- Error rate spikes above 5%.

UI/UX Requirements
As an Admin, I want to:

- See a dashboard with all LBs and their statuses (Green/Yellow/Red).
- Click on an LB to view detailed server metrics in a table/graph.
- Toggle between algorithms (e.g., switch from Round Robin to Least Connections) without downtime.

Example Edge Cases
As an Admin, I want to:

- See a warning when adding a duplicate server IP.
- Receive an error if I try to delete an LB that's still handling active traffic.
- View historical metrics (last 24h) to analyze trends.

Non-functional Requirements

Technologies

Back-end	ASP.NET Core
Front-end	React + TypeScript
API	REST + SignalR
Database	PostgreSQL
Caching	Redis
Infrastructure	Docker + Kubernetes

Requirements

1. **Performance**:
   - Handle ≥ 1000 RPS on mid-range hardware
   - < 50ms latency for 95% of requests
2. **Security**:
   - HTTPS support (Let's Encrypt)
   - DDoS protection (rate limiting)
   - Admin panel authentication
3. **Scalability**:
   - Horizontal scaling (adding new nodes)
   - Geo-distributed server support (GSLB)

Risks and Challenges

1. **DDoS Attacks**
2. **Load Balancer Bottleneck**
3. **Configuration Errors**: Broken routing after updates.