YiANDNS Project Wiki

DNS Gateway · YiAN Cloud Network DNS Gateway

v1.0.3 · Release Codename "Marhaba"

This document is compiled from the v1.0.3 source tree and its actual behavior. It covers the architecture, the web admin console,
encrypted DNS, the YiANLink protocol, scheduling mathematics, the configuration reference, operations & monitoring, and the developer guide.

Table of Contents

Table of Contents 2
1. Project Overview 5
1.1 Introduction and Positioning 5
1.2 Feature Overview 5
1.3 Glossary 6
1.4 How to Read This Document 7
2. Quick Start 9
2.1 System Requirements 9
2.2 Installation 9
2.3 First Run and Initialization 9
2.4 Minimal Working Configuration 10
3. System Architecture 12
3.1 Architecture Overview 12
3.2 Modules and Responsibilities 12
3.3 Request Processing Pipeline 13
3.4 Technology Stack and Dependencies 14
4. Web Admin Console 16
4.1 Feature Overview 16
4.2 Pages and Modules 16
4.3 Common Operations 18
4.4 Accounts and Authentication 19
4.5 REST API Reference 19
5. Encrypted DNS (DoT / DoH / DoQ) 21
5.1 Protocol Comparison 21
5.2 Protocol Configuration 21
5.3 TLS and Certificate Management 22
5.4 Performance and Tuning 22
5.5 Client Onboarding Examples 23
6. The YiANLink Protocol (Proprietary) 24
6.1 Design Background and Goals 24
6.2 Protocol Specification 24
6.2.1 Frame Format and Data Structures 24
6.2.2 Handshake and Connection Setup 24

6.2.3 Encryption and Authentication 24
6.2.4 Transport 25
6.3 State Machine and Lifecycle 25
6.4 Interoperability with Standard Protocols 25
6.5 Versioning and Compatibility 25
6.6 Error Handling 25
6.7 Implementation Reference (client example) 26
7. Scheduling and Mathematical Models 27
7.1 Objectives and Design Philosophy 27
7.2 The Formulas 27
7.2.1 Definitions and Symbols 27
7.2.2 Physical Meaning and Derivation 29
7.2.3 When Each Formula Runs 31
7.3 Scheduling Decision Flow 32
7.4 Tuning Guide 32
7.5 Worked Examples 33
8. Configuration Reference 34
8.1 File Structure 34
8.2 Configuration Items 34
8.3 Typical Configurations 38
9. Operations and Monitoring 40
9.1 Logging and Diagnostics 40
9.2 Metrics 40
9.3 Troubleshooting FAQ 41
9.4 Upgrade and Rollback 42
10. Developer Guide 43
10.1 Repository Layout 43
10.2 Development Environment 43
10.3 Build, Test, Release 43
10.4 Contribution Guidelines 44
11. Security Notes 45
11.1 Security Model and Threat Assumptions 45
11.2 Known Limitations and Best Practices 46
Appendices 47
Appendix A. YiANLink Frame Quick Reference 47

Appendix B. Full Formula Derivations 47
Appendix C. CLI / API Quick Reference 48
Appendix D. Changelog 49
Appendix E. References 50

1. Project Overview

1.1 Introduction and Positioning

YiANDNS (the YiAN Cloud Network DNS Gateway, engineering name **DNS Gateway**) is an enterprise-grade DNS service system for home and corporate LANs. It accepts DNS queries from LAN clients, intercepts advertising, tracking and malicious domains through its filtering engine, accelerates responses through caching, and forwards the rest to configurable upstream DNS servers — plain UDP/TCP, encrypted DoT/DoH/DoQ, or the proprietary YiANLink protocol — all administered through a web console.

The problems it solves:

- **Unified control**: a single DNS exit for the entire LAN, centrally enforcing ad blocking, domain allow/block lists, rewrites, and per-client policy groups;
- **Encrypted egress**: where plaintext DNS is vulnerable to hijacking or eavesdropping, it provides encrypted upstreams and encrypted server endpoints over DoT/DoH/DoQ;
- **Resolution quality and resilience**: intelligent multi-upstream scheduling (Load Balancing / Trend Prediction / Parallel Queries) with circuit-breaker health checks that automatically steer traffic away from degraded nodes.

The deliverable is a single executable fat jar built on Java 21 (virtual threads): 100% pure Java with zero native dependencies, packaged once and running directly on any OS (including Alpine/musl) and any architecture (x86/ARM64, e.g. Raspberry Pi). The current release is **v1.0.3** under the release codename **"Marhaba"** (covering the 1.0.0 → 1.2.20 range).

1.2 Feature Overview

Area	Capabilities
Web Admin Console	Vue 3 + Element Plus + ECharts: dashboard, query log, access rules, system settings; PBKDF2 passwords + Bearer sessions
Multi-protocol	Plain UDP/TCP (default 53); DoT (RFC 7858, 853); DoH (RFC 8484, 443, HTTP/2 and HTTP/1.1); DoQ (RFC 9250, 853, pure-Java QUIC); proprietary YiANLink (TCP 6833)
Smart scheduling	Three upstream query modes: Load Balancing (round-robin), Trend Prediction (cost-energy function + Softmax + SWRR dynamic weights), Parallel Queries (race all healthy upstreams); Split DNS routing; circuit-breaker health checks with automatic fallback
Filtering engine	AdGuard-style rules (hosts lines, double-bar domain blocks, @@ exceptions, $important/$client=/$dnstype= modifiers, regexes), domain suffix tree, per-client IP/CIDR profiles, remote rule sources with atomic hot swap
Caching	Default **Frequency-Aware Dynamic-TTL** engine (TTL

	floats with per-domain access heat; negative caching; statistics); falls back to Caffeine W-TinyLFU via `-Ddns.cache.engine=caffeine`
Query log and stats	Daily-rotated JSONL files + in-memory ring buffer; 24h/7d stat windows, TopN domains, block ratio
Capacity self-protection	Predictive capacity control: a virtual-thread concurrency gate built on Taylor-expansion remainders, rejecting fast under traffic spikes to prevent collapse
Observability	In-memory metrics registry (counters/latency), per-protocol counts, active connections, JVM/system status sampling (pure JDK APIs)
Plugins	ServiceLoader SPI: embedded in the fat jar + external `plugins/` directory; request/response filters, upstream/cache/config/metrics providers
Portability	Zero native code: DoT/DoH ride on JDK SSLContext + Netty, DoQ on pure-Java QUIC (Kwik) — no BoringSSL/openssl

1.3 Glossary

Term	Definition
DoT	DNS over TLS (RFC 7858): DNS messages over TLS on TCP port 853, each message prefixed with a 2-byte length field
DoH	DNS over HTTPS (RFC 8484): DNS carried as HTTP body/parameter over HTTPS (application/dns-message)
DoQ	DNS over QUIC (RFC 9250): DNS over QUIC streams with ALPN `doq`, multiplexed by design
YiANLink	The project's proprietary application-layer encrypted transport: AES-256-GCM + pre-shared key + long-lived TCP, ferrying DNS messages only
PSK	Pre-Shared Key: the 256-bit symmetric key of YiANLink, configured in Base64
Frequency-aware cache	The default cache engine (FrequencyAwareCache): cache TTL floats with per-domain access heat (exponentially decayed frequency) — hot domains persist, cold ones expire quickly
SWRR	Smooth Weighted Round Robin (same algorithm as Nginx): smooths selection proportionally to weight, avoiding bursts on high-weight nodes
Cost energy J	The core function of Trend Prediction scheduling: a scalar cost combining RTT level, deterioration trend, acceleration jitter and dispersion; lower is better

Softmax	Maps per-node costs to normalized weights: W_i = exp(-0.5·J_i) / Σ exp(-0.5·J_k)
EMA	Exponential moving average, used to smooth RTT and other time series
ALPN	Application-Layer Protocol Negotiation: negotiates h2, doq, etc. during the TLS handshake
SNI	Server Name Indication: the target hostname carried in the TLS ClientHello
NXDOMAIN	Authoritative "no such domain" (rcode 3); negative caching stores it briefly
Negative caching	Caching NXDOMAIN for negativeTtlSeconds to blunt repeated queries for deleted domains
W-TinyLFU	Caffeine's admission+eviction hybrid built on Count-Min Sketch frequency estimation; used by the fallback engine (`-Ddns.cache.engine=caffeine`)
Split DNS	Domain routing: matched domains (exact/wildcard) bypass smart scheduling and go straight to a dedicated upstream
Circuit breaking	An upstream that fails repeatedly is disabled (open), then probed again after cooldown (half-open)
Half-open connection	A TCP connection whose peer vanished without FIN/RST; YiANLink reclaims these via a 60s read timeout
fat jar	The single-file executable jar produced by maven-shade-plugin, bundling all modules and third-party dependencies
reactor	One build unit in a Maven multi-module build; this project has 12 reactors = 1 parent POM + 11 code modules
Predictive gate	The predictive capacity-control component: virtual threads + a dynamically resized Semaphore admitting or rejecting queries based on a Taylor forecast

1.4 How to Read This Document

Reader	Suggested path	Goal
Home / small-team user	Chapter 2 → 4.1–4.3 → 5.5	Deploy, drive the console, connect clients
Operations engineer	Chapter 8 → Chapter 9 → 5.2–5.4 → 11.2	Tune configuration, manage certificates, monitor and troubleshoot, harden
Backend developer	Chapter 3 → Chapter 7 → Chapter 10 → Appendices A/B	Understand the architecture and the mathematics, contribute
Security auditor	Chapter 11 → 5.3 → 6.2/6.6 → 4.4	Threat model, cryptography usage, known limitations

Protocol integrator	Chapter 6 → Appendix A → Chapter 7	Implement a YiANLink client, understand scheduling behavior

---

2. Quick Start

2.1 System Requirements

Item	Requirement
Runtime	JDK 21 or later (OpenJDK / IBM Semeru / Huawei BiSheng all work; built with `--release 21`)
Memory	≥ 512 MB (1 GB recommended; the default 100k-entry cache costs tens of MB of heap)
Disk	≥ 100 MB (plus query-log growth per retention policy)
OS	Any (Windows / Linux / macOS, incl. Alpine/musl containers; x86 and ARM64)
Network	Data plane UDP+TCP 53 (configurable); admin plane TCP 8080 (intranet/localhost recommended); optional 853/443/6833
Privileges	On Linux, binding 53 requires root or `setcap 'cap_net_bind_service=+ep' <jdk>/bin/java`; on Windows, allow the ports in the firewall
Optional	Certificate files (required to enable DoT/DoH/DoQ); a YiANLink PSK (required to enable YiANLink)

2.2 Installation

Option A — download a release (recommended)

1. Download `dns-proxy-1.0.3.jar` from the Releases page of the project's hosting platform (Gitee);

2. Place it in any deployment directory (e.g. `/opt/yiandns/`); that directory becomes the base for resolving `data/` and `TLS/`.

> *The `.gitignore` excludes `target/` and `*.jar` — source distributions contain no build artifacts, so a `git clone` must be built locally.*

Option B — build from source

Prerequisites: JDK 21+ (with `JAVA_HOME`), Maven 3.6+ (global `mvn`).

```
# Windows (from the project root)
powershell -File "build.ps1" clean package
# Artifact: dns-proxy\target\dns-proxy-1.0.3.jar
# Linux / macOS
mvn -s maven-settings.xml -B clean package
```

After building, run the end-to-end smoke test (starts UDP+TCP on random ports, 9 assertions):

```
powershell -File "smoke.ps1"
```

2.3 First Run and Initialization

```
# Start with the example config (DNS on 0.0.0.0:53, admin on 127.0.0.1:8080)
java -jar "dns-proxy-1.0.3.jar" "config\config.example.yaml"
```

- Config file resolution: explicit path argument → `config.yaml` / `config.yml` / `config.json` in the working directory → built-in defaults if none exist;

- Layered loading: built-in defaults → user file (deep merge) → runtime overlay (web-saved settings); unspecified fields fall back to defaults;
- The startup banner prints the version (e.g. `DNS 服务启动完成 (v1.0.3)`), each listener, the cache engine (by default 启用[频率感知] — frequency-aware), and the predictive-gate parameters.

First login: open `http://127.0.0.1:8080` and sign in with the default credentials `admin` / `admin`.

> ⚠ ***Security warning**: the default credentials are public knowledge. Before exposing the service to a public or untrusted network, change `web.adminPassword` and restrict `web.bindAddress` to a trusted segment.*

Verify the DNS service:

```
nslookup baidu.com 127.0.0.1
```

Graceful shutdown: pressing `Ctrl+C` in the foreground triggers the shutdown hook, which stops in order: web admin → DoQ/DoH/DoT → UDP/TCP listeners → drain in-flight queries (500 ms by default) → plugins → upstream connection pools → flush the query-log buffer.

Exit codes:

Code	Constant	Meaning
0	EXIT_OK	Normal exit
1	EXIT_STARTUP_FAILURE	Startup failed (no valid upstreams, missing certificate, unexpected exception, …)
2	EXIT_CONFIG_ERROR	Configuration validation failed (each field path and reason is printed)
3	EXIT_PORT_IN_USE	A DNS/web port is already in use

Fault tolerance: a missing rules file is skipped with a warning; a single failing plugin never blocks startup; an unwritable query-log directory degrades to memory mode; unreachable upstreams are handled at runtime by the circuit breaker + failover, never by refusing to start.

(Optional) systemd unit for Linux:

```
# /etc/systemd/system/yiandns.service
[Unit]
Description=YiANDNS DNS Gateway
After=network.target

[Service]
ExecStart=/usr/bin/java -jar /opt/yiandns/dns-proxy-1.0.3.jar /opt/yiandns/config.yaml
WorkingDirectory=/opt/yiandns
SuccessExitStatus=143
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

2.4 Minimal Working Configuration

```
server:
  bindAddress: 0.0.0.0
  udpPort: 53
  tcpPort: 53
upstreams:
  - address: 223.5.5.5
    protocol: UDP
```

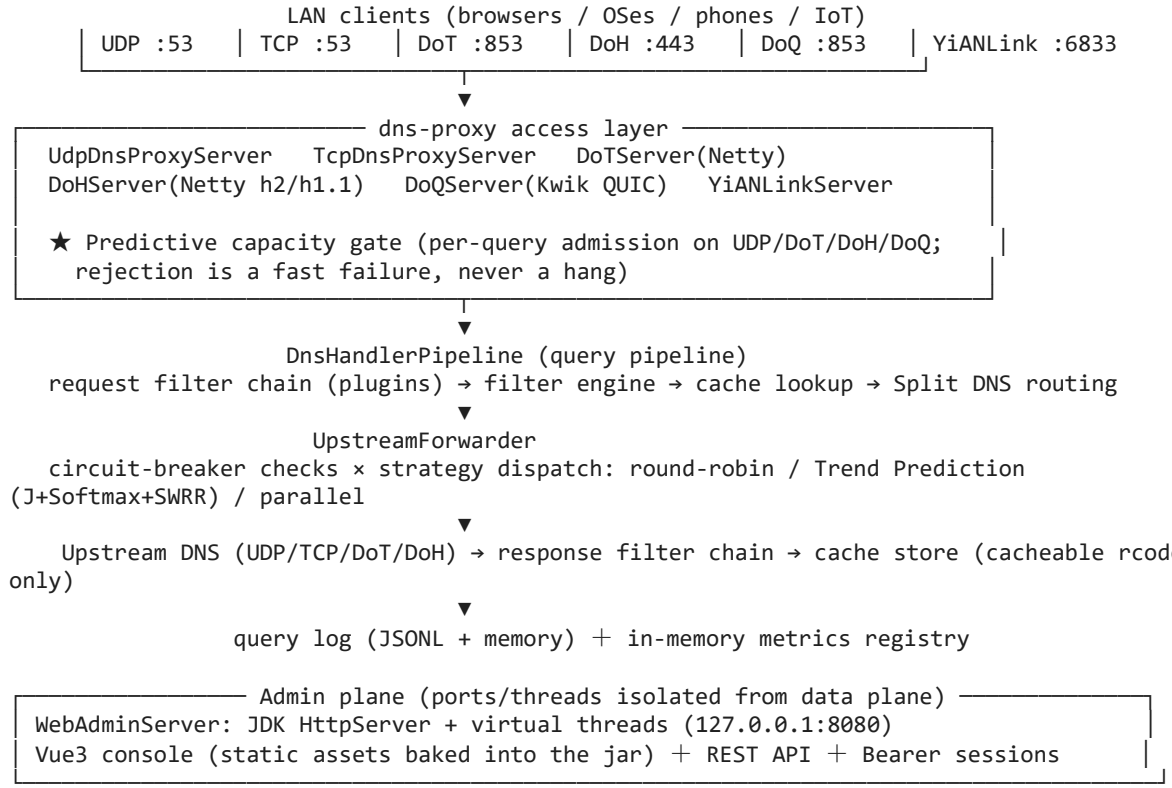
```
  upstream:
    strategy: ROUND_ROBIN
  cache:
    enabled: true
  web:
    enabled: true
    bindAddress: 127.0.0.1
    port: 8080
    adminUsername: admin
    adminPassword: "change-me-to-a-strong-password"
```

Save the above as `config.yaml` next to the jar and start with `java -jar dns-proxy-1.0.3.jar`.

---

3. System Architecture

3.1 Architecture Overview



Data-flow notes: a failure in the admin plane never affects the query path; every data-plane entry point that spawns a per-query virtual thread passes through the predictive gate; TCP and YiANLink use one thread per connection with serialized queries inside a connection, so they are governed at the connection level rather than by the per-query gate.

3.2 Modules and Responsibilities

The Maven reactor has 12 build projects = 1 parent POM (aggregation + dependency management) + 11 code modules:

Module	Responsibility
dns-server-api	DNS message/exchange/codec contracts (DnsRequest/DnsResponse/DnsExchange/DnsCodec)
dns-core	dnsjava-backed codec (DnsjavaCodec) and answer factory (NOERROR/NXDOMAIN/REFUSED/CNAME/A/AAAA)
plugin-api	Extension points: DnsPlugin, request/response filters (sealed decisions), upstream/cache/config/metrics providers
config	Layered YAML/JSON loading (ConfigNode deep merge), field-level validation (aggregated ConfigError), hot-reload listeners, UpstreamPolicy

logging	Query log: daily-rotated JSONL / memory ring buffer, async flushing, 24h stat buckets
metrics	In-memory metrics registry (counters/latency/gauges, QPS tick)
filter-engine	AdGuard-style rule parsing, domain suffix tree, allow/block/rewrite matching, per-client engine (UserAwareFilterEngine)
upstream-resolver	UDP/TCP/DoT/DoH transports, three strategy selectors, Trend Prediction package (cost energy + SWRR), circuit-breaker health, Split DNS forwarding, upstream address parsing (UpstreamAddressParser)
cache	Frequency-aware dynamic-TTL cache (default engine: heat-floating TTL, negative caching, statistics); falls back to Caffeine W-TinyLFU via `-Ddns.cache.engine=caffeine`
dns-proxy	UDP/TCP/DoT/DoH/DoQ/YiANLink servers, the query pipeline, the predictive gate (concurrent package), entry point DnsGatewayApplication
web-admin	Admin REST API (JDK HttpServer + virtual threads, PBKDF2 passwords, Bearer sessions, Vue3 console static assets)

3.3 Request Processing Pipeline

Assembly at startup (12 steps; a failure rolls back in reverse order):

1. Load and strictly validate configuration (DnsConfigManager.load; validation failure exits with code 2)
2. In-memory metrics registry
3. Query-log service (falls back to memory mode with a WARN if files fail)
4. Plugin loading (classpath + plugins/; a single failure is skipped with a warning)
5. Dynamic rule service RuleService (local TSV + remote rule sources)
6. Filter engine (UserAwareFilterEngine, hot rebuilds on rule changes)
7. Cache (frequency-aware by default; explicit `caffeine` fallback; disabled entirely when enabled=false)
8. Upstream forwarder (candidate construction + strategy selectors + Trend Prediction scheduler on demand)
9. Query pipeline DnsHandlerPipeline (filter → cache → upstream orchestration)
10. UDP/TCP listeners
11. Secure transports DoT/DoH/DoQ (certificates loaded once, shared + hostname validation) and the YiANLink listener; the predictive gate is a global singleton
12. Web admin plane

Query path (UDP example):

1. The receive thread gets the datagram and parses the DNS wire format;

2. Predictive gate `executeTask`: if `tryAcquire` fails, no thread is spawned — SERVFAIL is answered directly on the accept thread (DoH returns HTTP 200 + DNS SERVFAIL; DoQ resets the stream per RFC 9250 §4.3);

3. Inside the virtual thread: decode → request filter chain (plugins) → filter engine (order: important blocklist → allowlist/exceptions → rewrites → normal blocklist → allow);

4. Cache lookup (key = normalized name|qtype): on hit, rewrite the requestId, mark fromCache, and return immediately;

5. On miss → Split DNS match: if matched, go straight to the dedicated upstream (its own 2000 ms timeout; SERVFAIL on timeout; circuit breaker falls back to the default engine); otherwise dispatch by strategy;

6. UpstreamForwarder tries candidates per strategy (round-robin / SWRR weight order / race all); SERVFAIL/REFUSED/FORMERR/NOTIMP are retryable and move to the next candidate; NXDOMAIN/NOERROR are terminal; transport errors increment the circuit-breaker counter;

7. Response filter chain (plugins) → cacheable rcodes (NOERROR/NXDOMAIN only) are stored with their min TTL; CNAME rewrites additionally resolve the target and attach its A/AAAA records in the same answer (falling back to CNAME-only if that resolution fails);

8. Query log (async) + metrics (dns.query.*, dns.protocol{...}, latency) → reply (UDP written back to the source; TCP/DoT framed; DoH 200 application/dns-message; DoQ written on the same stream then FIN).

3.4 Technology Stack and Dependencies

Dependency	Version	Purpose
Java	21 (virtual threads, records, sealed types)	Runtime and language baseline
Maven	3.6+ (maven-shade-plugin 3.6.0)	Build and fat jar
dnsjava	3.6.1	DNS codec and record model
Netty	4.1.115.Final	DoT (NIO+TLS) and DoH (HTTP/2 + HTTP/1.1) server transport; TLS via JDK SSLContext, no tcnative
Kwik	0.8.6 (package net.luminis.quic)	Pure-Java QUIC implementation carrying DoQ
Caffeine	3.1.8	Fallback cache engine (-Ddns.cache.engine=caffeine; W-TinyLFU)
SnakeYAML	2.3 (SafeConstructor)	YAML config parsing
Jackson	2.18.2	JSON config parsing and REST JSON serialization
BouncyCastle	bcpkix-jdk18on 1.78.1	PEM private-key parsing (PKCS#8/PKCS#1/SEC1)
JUnit	5.11.3	Unit testing
Spring Boot	none	The project contains no Spring; the web layer is JDK HttpServer +

		virtual threads

---

4. Web Admin Console

4.1 Feature Overview

The console lives at `http://<web.bindAddress>:<web.port>/index.html` (default `http://127.0.0.1:8080`). Top navigation:

Page	Function
Dashboard	Four stat cards (current QPS / total queries / blocked count / cache hit ratio) + four ECharts (top requested domains, top blocked domains, protocol share, upstream counts & latency)
Query Log	Recent queries (domain/type/result/source protocol/client/upstream/latency/matched rule), details, statistics & TopN, clear log/stats
Access Rules	Five independently saved rule groups (blocklist / allowlist / Split DNS / rewrites / custom), hot-effective on save; remote rule source management (auto/manual updates)
Settings	Basic (system status, data storage, system operations, logging), DNS Service (upstreams, cache, upstream query mode), DNS Protocols (UDP/TCP, encrypted DNS, YiANLink), upstream server management
Login/Account	Single-administrator login, logout, automatic 401 sign-out

Every admin API (except login and static pages) requires `Authorization: Bearer <token>`; on 401 the front-end clears its local token and returns to the login page.

4.2 Pages and Modules

4.2.1 Login: username + password (`POST /api/auth/login`, anonymous). On success the token is stored in localStorage (key `dns_admin_token`, valid 8 hours).

4.2.2 Dashboard: four stat cards (big number + mini trend) and four charts fed by `GET /api/metrics`, `GET /api/stats?topN=5`, `GET /api/status`, refreshed on a timer; when the backend has no data the UI shows a genuine empty state (never mock data).

4.2.3 Query Log: served from the in-memory ring buffer (up to logging.maxEntries); supports recent-N pulls, per-record details, and TopN domains within the stats window (statsRetentionHours); maintenance actions "clear query log / clear statistics" (`/api/maintenance/*`).

4.2.4 Access Rules (core):

- Five rule tabs, each saved independently and hot-rebuilding the filter engine on save (atomic swap; lock-free queries):
  - **Blocklist**: domains to block (double-bar Adblock rules, hosts 0.0.0.0 lines, bare domains, `*.wildcards`, …);

  - **Allowlist**: @@ exception rules, which take precedence over blocklists;
  - **Split DNS**: exact/wildcard domain → dedicated upstream (see 8.2 domainRouting); saving validates that the target is not the gateway's own listening address (loop prevention);
  - **Rewrites**: source domain → target IP/IPv6/CNAME (CNAME loops detected automatically);
  - **CNAME rewrite chaining**: the answer returns the CNAME and additionally resolves the target, attaching its A/AAAA records in the same reply (CNAME-only if that fails) — clients get an IP without a second query;
  - **Custom**: mixed group, auto-classified per line (@@ → allowlist, has-target → rewrite, otherwise → blocklist).
- **Remote rule sources**: each group can subscribe to HTTP(S) feeds (a four-column table of name/URL/rule count/last update, 10 rows per page); feeds are fetched concurrently on an interval with automatic format detection (Adblock/hosts/bare domains/mixed); manual "update now" and auto-update off-switch (intervalSeconds=0) supported; local rules are edited in the tab's input area, not inside the sources table.

4.2.5 Settings:

- **Basic**:
  - System status (3-second partial refresh): uptime, thread count, Load1/5/15 (N/A on Windows), heap/stack/direct memory, YoungGC/OldGC counts and durations, recent GC pause, current OS (os.name/version/arch) and a three-line Java version block (version/vendor/runtime/vm); every metric is collected with stock JDK APIs;
  - Data storage: `storage.dataDir` (the base for query-log persistence; relative paths resolve against the jar directory);
  - System operations: restart / shutdown (async 202, executed ~300 ms later; a restart reassembles everything and invalidates login sessions);
  - Logging: retention hours, stats window, storage type (JSONL/MEMORY), level.
- **DNS Service**:
  - Upstreams: add/remove/edit entries (address accepts host / host:port / full URL + protocol UDP/TCP/DOT/DOH), the **upstream query mode** dropdown (1 Load Balancing / 2 Trend Prediction / 3 Parallel Queries, hot-updated), the resetStateOnUpdate switch and the "reset upstream state" button;
  - Cache: switch, maxEntries, defaultTtlSeconds, negativeTtlSeconds, minTtlSeconds, maxTtlSeconds.
- **DNS Protocols**:
  - UDP/TCP switches and ports;
  - Encrypted DNS (DoT/DoH/DoQ): per-protocol switches and ports, certificate paths (certFile/keyFile), external hostname, minimum TLS version (DoH), the legacyRsaCompat compatibility switch;
  - YiANLink: switch, port (default 6833), PSK (random generation, Base64 32-byte validation); saving writes the top-level `yianlink` block and prompts for a restart.
- **Upstream Servers**: node management and connection status.

4.3 Common Operations

Screenshots should be captured from your actual deployment; the steps below follow the default layout.

Operation 1 — change upstreams and switch query mode (hot-effective, no restart)

1. Open Settings → DNS Service → Upstreams;

2. Add/remove/edit entries (the address accepts host / host:port / a full URL such as `https://doh.pub/dns-query` or `tls://dot.pub`; protocols UDP/TCP/DOT/DOH, and the URL scheme must agree with the protocol);

3. Pick Load Balancing / Trend Prediction / Parallel Queries in the query-mode dropdown;

4. Save: the whole config is atomically validated → written back to the startup config file → applied immediately at runtime. On failure nothing is applied and the validation errors are shown.

Operation 2 — add a blocking rule

1. Open Access Rules → Blocklist;

2. Paste rules (e.g. the double-bar block `||ads.example.com^`, a hosts line `0.0.0.0 tracker.net`, a wildcard `*.bad.io`);

3. It takes effect immediately; the dashboard's blocked counter rises, and the matched rule appears in the query log's matchedRule field.

Operation 3 — deploy a certificate and enable DoH

1. Place the PEM full chain and private key at `TLS/fullchain.pem` and `TLS/privkey.pem` in the runtime directory;

2. Settings → DNS Protocols → Encrypted DNS: tick DoH, port 443, path `/dns-query`; set server.hostname (e.g. `dns.example.com`; validated at startup against the certificate SAN/CN);

3. Save and restart; the banner prints `DoH → https://dns.example.com/dns-query` as the client hint;

4. Verify from a client: point Firefox/Chrome DoH at that URL, or run `kdig +https dns.example.com example.com`.

Operation 4 — enable the YiANLink private channel

1. Settings → DNS Protocols → YiANLink: switch it on, keep port 6833;

2. Generate a PSK ("random") or paste a 256-bit Base64 key, and distribute it to trusted clients;

3. Save (a restart is prompted) → restart → the banner shows the YiANLink listener;

4. Clients connect per the Chapter 6 spec; the query log's source protocol shows YiANLink.

Operation 5 — configure Split DNS

1. Access Rules → Split DNS;

2. Add an entry: domain pattern (`www.a.com` exact, or `*.b.com` wildcard matching subdomains only) → dedicated upstream (e.g. `udp://119.29.29.29:5353`) → optional timeout (default 2000 ms);

3. Hot-effective; matched queries bypass smart scheduling, and if the dedicated upstream is unreachable the circuit breaker falls back to the default engine.

Operation 6 — change the admin password

1. Edit `web.adminPassword` in the config file (or save via Settings) → restart; all old sessions are invalidated and you re-login with the new password.

4.4 Accounts and Authentication

Item	Details
Account model	Single administrator (web.adminUsername / web.adminPassword); no multi-user or RBAC
Password storage	PBKDF2WithHmacSHA256, 120k iterations + random salt, stored as `pbkdf2$iter$saltB64$hashB64`
Verification	MessageDigest.isEqual constant-time comparison, resisting timing side channels
Sessions	Login returns a 32-byte SecureRandom token valid 8 hours; logout invalidates it
Transport auth	All REST APIs (except login and static pages) require `Authorization: Bearer <token>`; 401 auto-signout
Network isolation	web.bindAddress defaults to 127.0.0.1 and is separate from the DNS data plane; the admin thread pool is independent, so its failure never affects queries
Sensitive data	`GET /api/config` masks sensitive fields

4.5 REST API Reference

Everything except `POST /api/auth/login` and static pages requires Bearer auth.

Method	Path	Description
POST	/api/auth/login	Anonymous login; returns a token (8-hour validity)
POST	/api/auth/logout	Invalidate the current session
GET	/api/auth/me	Current session info
GET	/api/status	DNS/web ports and running state, cache size, etc.
GET	/api/connections	Active connection list (TCP/DoT/DoH/YiANLink)
GET	/api/queries?limit=N	Recent query log
GET	/api/queries/detail	Query log details
GET	/api/stats?topN=N	Query statistics and top domains
GET	/api/rules?group=	Read a rule group's raw text, counts and source status (group=blocklist/allowlist/rewrite/custom/domainRouting)
POST	/api/rules	Append rule text (body: content + group); persists to data/filters/user-rules.tsv and hot-rebuilds asynchronously
PUT	/api/rules	Replace an entire rule group with text
GET	/api/rules/sources	List remote HTTP(S) rule sources and their download status
POST	/api/rules/sources	Add a remote source (name + url + group + intervalSeconds + enabled)
PUT	/api/rules/sources	Replace a group's source list (intervalSeconds=0 disables auto-update)

DELETE	/api/rules/sources?id=	Delete a remote source and clear its cache
POST	/api/rules/sources/refresh[?id = or ?group=]	Refresh one / a group's / all remote sources now
GET	/api/config	Configuration tree (sensitive fields masked)
PUT	/api/config	Incremental patch (deep merge, atomic whole-file validation); on success written back to the startup config file, surviving restarts
GET	/api/metrics	Metric counters and average latency
GET	/api/system/info	System status sampling (uptime/load/memory/GC/threads/OS/Java)
POST	/api/system/restart	Asynchronously restart DNS/web services (re-read config, re-bind ports)
POST	/api/system/shutdown	Asynchronously shut down and exit the process
POST	/api/maintenance/clear-query-log	Clear the query log
POST	/api/maintenance/clear-stats	Clear statistics
POST	/api/upstream/reset	Manually reset upstream circuit-breaker and RTT state (pairs with upstream.resetStateOnUpdate)
GET	/api/plugins	Loaded plugins
GET	/index.html	Admin page (anonymous; SPA static assets under /assets/* are also anonymous)

Config-save semantics: `PUT /api/config` validates the whole configuration first; on success it is **atomically written back to the startup config file** (YAML/JSON by extension) and then applied to the runtime. If writing to disk fails, nothing is applied — settings can never be silently lost on restart. When multiple files were merged at startup, the first one is written back.

---

5. Encrypted DNS (DoT / DoH / DoQ)

5.1 Protocol Comparison

Dimension	DoT	DoH	DoQ
Spec	RFC 7858	RFC 8484	RFC 9250
Transport	TLS over TCP	HTTPS (this project: h2 and HTTP/1.1)	QUIC (ALPN doq, TLS 1.3 built-in)
Default port	853	443	853
Framing	2-byte big-endian length prefix	HTTP body (POST) or ?dns= base64url (GET)	One query per bidirectional QUIC stream, 2-byte length prefix
Connection model	Long-lived connection, many queries	keep-alive / h2 multiplexing	Stream-level concurrency, no head-of-line blocking
Typical use	OS-level DNS (Android Private DNS, systemd-resolved)	Browsers (Firefox/Chrome), embedded clients	Latency- and concurrency-sensitive clients
This project	Netty NIO + JDK SSLContext; **strict TLS 1.2 only, ALPN fully disabled**; 30s idle close	Netty HTTP/2 (Http2FrameCodec/MultiplexHandler) + HTTP/1.1 fallback pipeline; non-whitelisted ALPN is explicitly rejected	Kwik pure-Java QUIC; ServerConnector + doq protocol factory; may share port 853 with DoT
Certificate	TLS/fullchain.pem + privkey.pem (shared by all three)	same	same; **RSA private key only (PKCS#8)**

Additional port-validation rules: dot/doh overlapping tcpPort/udpPort is an error; doq overlapping udpPort is an error; dot and doh on the same port is an error (**dot+doq both on 853 is legal** — RFC-sanctioned sharing); doh.path must match `^/[a-zA-Z0-9_\-/]*$`.

5.2 Protocol Configuration

```
server:
  hostname: dns.example.com        # optional: external name, validated against the cert at
startup, echoed in the banner
  tls:                             # shared by all three protocols (required when any is
enabled)
    certFile: TLS/fullchain.pem    # PEM full chain
    keyFile: TLS/privkey.pem       # PEM private key (PKCS#8 or PKCS#1; RSA only for DoQ)
    minVersion: TLSv1.2            # affects DoH only (TLSv1.2/TLSv1.3); DoT is fixed
TLS1.2; DoQ is fixed TLS1.3
    legacyRsaCompat: false         # true also offers TLS_RSA_* (legacy clients;
SSLv3/TLSv1/1.1 stay blocked)
  dot:
    enabled: true
    port: 853
  doh:
    enabled: true
    port: 443
```

```
    path: /dns-query
  doq:
    enabled: true
    port: 853                       # may share the port with dot
```

- Certificate paths are mandatory once any protocol is enabled; a failed load / a key that does not match the certificate / a hostname not covered → startup fails with a precise reason (listing the domains the certificate actually covers);
- The banner prints each listener and a client hint (e.g. `DoT → dns.example.com:853 (DOT)`);
- Settings → DNS Protocols edits all of this and writes it back to the config file; port changes re-bind listeners and require the restart the UI prompts for.

5.3 TLS and Certificate Management

Requirements:

- PEM full chain (server certificate + intermediates in order) + PEM private key;
- Private keys may be PKCS#8 / PKCS#1 / SEC1 (parsed via BouncyCastle); **DoQ (Kwik) accepts RSA keys only, in PKCS#8**;
- At load time a "random-challenge signature + certificate-public-key verification" pairing check runs; a mismatch refuses startup.

Obtaining a CA certificate (Let's Encrypt example):

```
certbot certonly --standalone -d dns.example.com
# Produces /etc/letsencrypt/live/dns.example.com/fullchain.pem and privkey.pem
# Copy them into TLS/ (or configure absolute paths directly)
```

Self-signed certificates (intranet):

```
openssl req -x509 -newkey rsa:2048 -nodes -days 3650 \
  -keyout TLS/privkey.pem -out TLS/fullchain.pem \
  -subj "/CN=dns.lan" \
  -addext "subjectAltName=DNS:dns.lan,IP:192.168.1.2"
```

Clients must explicitly trust self-signed certificates (import into the OS/browser trust store, or use tool flags that skip verification). **The SAN must include the hostname or IP clients actually use**, otherwise mainstream clients (JDK HttpClient, browsers) reject the handshake on hostname verification.

Renewal: replace the files under `TLS/` → restart (certificates load once at startup today). Enabling `legacyRsaCompat` needs a restart to take full effect; disabling takes effect immediately.

Hostname validation: with `server.hostname` set, startup verifies that the certificate's SAN dNSName entries (single-level wildcards like `*.example.com` supported, falling back to CN when no dNSName exists) cover it; otherwise startup fails and lists the domains actually covered. The banner also prints DoH/DoT/DoQ client hints.

5.4 Performance and Tuning

Benchmark (Raspberry Pi 4B / ARM64 / Huawei BiSheng JDK 21, 1200 concurrent connections over the public internet):

Protocol	QPS	Success rate
UDP	6599+	95.6%
TCP	3974+	99.2%

Memory and GC stayed flat during the runs — no OOM, minimal pauses; under the same conditions connection stability beats mainstream open-source peers.

Tuning tips:

- Encrypted protocols pay a TLS-handshake cost: enable session resumption / long-lived connections client-side; the server deliberately tightens its session cache (a hardening choice) — do not loosen it;
- DoT is fixed to TLS 1.2 (a compatibility trade-off); for TLS 1.3 semantics use DoH (minVersion=TLSv1.3) or DoQ;
- Under high concurrency watch the predictive-gate parameters (7.4) and `server.threads`; prefer h2 multiplexing for DoH clients to reduce connection counts;
- When 443 is shared with other web services, prefer a dedicated IP/port to avoid ALPN/certificate conflicts;
- Upstream side: mixing encrypted (DOT/DOH) with plain upstreams works well — Trend Prediction tilts traffic automatically by RTT.

5.5 Client Onboarding Examples

Client	How to connect
Windows nslookup (plain)	`nslookup baidu.com 127.0.0.1`
Android (DoT)	Settings → Network → Private DNS → enter `dns.example.com` (needs a trusted certificate + SAN match)
systemd-resolved (DoT)	`/etc/systemd/resolved.conf`: `DNS=192.168.1.2` + `DNSOverTLS=yes`
Firefox (DoH)	Settings → Privacy & Security → DNS over HTTPS → custom URL `https://dns.example.com/dns-query`
Chrome (DoH)	Settings → Security → Use secure DNS → custom provider with the same URL
kdig (DoT/DoH/DoQ)	`kdig +tls @dns.example.com -p 853 example.com`; `kdig +https @dns.example.com example.com`; `kdig +quic @dns.example.com -p 853 example.com`
curl (DoH)	`curl --doh-url https://dns.example.com/dns-query https://example.com`
YiANLink client	Implement per Chapter 6 (PSK + TCP 6833)

DoT failing while DoQ works usually means the certificate is untrusted or the address is outside the SAN (DoQ tools often skip verification); the server itself never bypasses hostname checks.

---

6. The YiANLink Protocol (Proprietary)

6.1 Design Background and Goals

DoH/DoT depend on the public PKI: issuing and rotating certificates in intranet or offline environments is painful, and TLS fingerprints are conspicuous and easy to identify or hijack. YiANLink is the project's **proprietary application-layer encrypted transport**, designed to:

- **Eliminate certificate operations**: a 256-bit PSK replaces PKI entirely — deploy and go;
- **Resist identification**: custom binary frames + AEAD encryption produce weak traffic fingerprints;
- **Defeat replay**: an in-frame timestamp validated against a 60-second receive window;
- **Stay single-purpose**: inside the gateway it **only ferries encrypted DNS bytes** — no parsing, caching or filtering (all of that stays in the unified pipeline); the protocol layer is fully decoupled from business logic via the callback-style YiANLinkHandler;
- **Multiplex without head-of-line blocking**: clients send through a rotating connection pool instead of serializing on one socket.

6.2 Protocol Specification

6.2.1 Frame Format and Data Structures

Transport frame (on top of TCP, one frame per message):

```
[4B Length(big-endian)] [12B IV] [Ciphertext(N)] [16B GCM Tag]
Length = 12 + N + 16 (everything after the Length field)
Ciphertext = AES-256-GCM(PSK, IV, plaintext), no additional AAD
```

Frame plaintext:

```
[8B timestamp(big-endian, Unix ms)] [4B StreamId(big-endian)] [payload]
payload = a complete DNS message (wire format, no TCP length prefix)
```

6.2.2 Handshake and Connection Setup

There is no dedicated handshake: the client connects and sends its first frame; the server decrypts and verifies on arrival (PSK gives implicit authentication) —

1. TCP three-way handshake completes → the connection is READY;
2. The client sends a query frame (random IV; timestamp + StreamId + DNS query);
3. The server decrypts → verifies the Tag → checks the timestamp window → invokes `onDnsPayloadReceived(payload, clientIp)` into the unified pipeline;
4. The answer is framed and written back with the **original StreamId**;
5. The connection carries many request/response round trips (long-lived reuse).

6.2.3 Encryption and Authentication

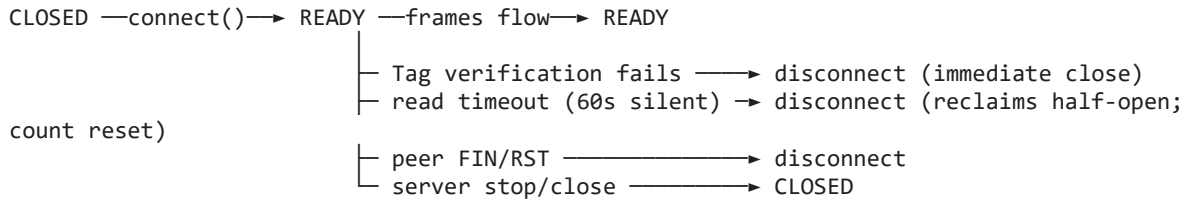
- Cipher: AES-256-GCM (AEAD); the key is the PSK (configured as a Base64-encoded 32-byte secret);
- IV: 12 random bytes per frame — **reuse is strictly forbidden** (IV reuse under one key breaks GCM's security);
- Integrity: the 16-byte GCM tag; any tampering fails decryption;

- Anti-replay: the plaintext carries the sender's timestamp; the receiver requires ≤ 60 seconds of clock skew.

6.2.4 Transport

Long-lived TCP (default port **6833**, configurable; must not collide with plain TCP/DoT/DoH ports). UDP/QUIC carriage is not supported in this version; the server runs one virtual thread per connection, processing frames sequentially within it.

6.3 State Machine and Lifecycle



- **Half-open governance**: when a client vanishes without FIN, the 60s read timeout reclaims the connection and zeroes its active count (never relying on FIN/RST; covered by regression tests);
- open/close are strictly paired on the connection thread, keeping `/api/connections` and monitoring accurate;
- On shutdown the server closes each connection, following the Gateway's unified reverse-order rollback.

6.4 Interoperability with Standard Protocols

- Coexistence: YiANLink listens alongside UDP/TCP/DoT/DoH/DoQ without interference;
- One pipeline: YiANLink queries enter the same DnsHandlerPipeline — filtering, caching, Split DNS and scheduling behave identically to every other protocol;
- Client semantics: the real TCP peer IP flows through the entire chain (IP allow/block lists, rate limiting, the query log's client field, the `dns.protocol{YIANLINK}` counter);
- Monitoring contract: counter key `dns.protocol{YIANLINK}` ⟷ display name `YiANLink`; active connections roll into totalActiveConnections (yianLinkConnections).

6.5 Versioning and Compatibility

- Current protocol version v1 (the frame layout above carries no version-negotiation field); a future v2 would add a version byte while keeping a v1 compatibility window;
- Rotating the PSK requires synchronizing the server and every client; old and new keys are mutually incompatible during the transition;
- Port changes take effect after a restart (the UI already prompts this).

6.6 Error Handling

Condition	Server behavior
GCM Tag verification fails (wrong key / tampering)	Disconnect immediately (CryptoException → close)
Timestamp skew > 60 s (replay / clock drift)	Silently drop the frame — no disconnect, no reply
Invalid frame length / truncated stream	Disconnect
enabled=true but PSK missing	Startup fails (config validation)

Port conflict	Startup fails (exit-code 3 semantics, PortInUse)
60 s of silence (half-open / idle)	Read timeout reclaims the connection

6.7 Implementation Reference (client example)

Python reference (requires the `cryptography` package):

```
import os, socket, struct, time, base64
from cryptography.hazmat.primitives.ciphers.aead import AESGCM

PSK = base64.b64decode("<the Base64 key from config>")   # must decode to 32 bytes
key = AESGCM(PSK)
sid = 0

def build_frame(dns_wire: bytes) -> bytes:
    global sid
    sid += 1
    plain = struct.pack(">Q", int(time.time() * 1000)) + struct.pack(">I", sid) + dns_wire
    iv = os.urandom(12)                                # fresh IV per frame; never reuse
    ct = key.encrypt(iv, plain, None)
    return struct.pack(">I", len(iv) + len(ct)) + iv + ct

def parse_frame(buf: bytes) -> tuple[int, bytes]:
    total = struct.unpack(">I", buf[:4])[0]
    iv, body = buf[4:16], buf[16:4 + total]
    plain = key.decrypt(iv, body, None)
    ts, resp_sid, payload = plain[:8], plain[8:12], plain[12:]
    assert abs(int.from_bytes(ts, "big") - int(time.time() * 1000)) <= 60_000
    return int.from_bytes(resp_sid, "big"), payload   # answers reuse the original StreamId

sock = socket.create_connection(("192.168.1.2", 6833))
sock.sendall(build_frame(dns_query_wire))
# read a 4B length + body → parse_frame → check the StreamId matches the request
```

Server side: the `com.example.dns.proxy.yianlink` package of `dns-proxy` (YiANLinkServer / YiANLinkHandler); the business callback signature is `onDnsPayloadReceived(byte[] payload, InetSocketAddress clientIp)` (the two-argument form; the old single-argument signature is deprecated).

7. Scheduling and Mathematical Models

7.1 Objectives and Design Philosophy

Upstream scheduling pursues four goals: **low latency** (favor low-RTT nodes), **high availability** (isolate degraded or failed nodes via circuit breaking), **resilience** (automatic failover and fallback), and **fairness** (round-robin splits traffic evenly).

The "upstream query mode" exposes three options (hot-changeable under Settings → DNS Service):

Option	Config value	Semantics	Mathematical model
1 Load Balancing	ROUND_ROBIN	Rotate one upstream at a time; circuit-broken nodes always act as the last resort, and the cursor rotates only over the healthy subset (mixed breaker states eliminate first-pick bias; with everything healthy it matches classic round-robin exactly)	none (deterministic rotation)
2 Trend Prediction	TREND_PREDICTION	Weights recomputed from each node's RTT time series, then applied via SWRR	cost-energy function J + Softmax + integrated health window + SWRR (7.2.2)
3 Parallel Queries	PARALLEL	Query all healthy upstreams concurrently, take the first valid answer; maxUpstreamAttempts limits participation (0 = all)	none (a race)

Node weights are computed entirely by the scheduler — **no manual weights are needed (or supported)**. When Split DNS matches, the query bypasses this machinery and goes straight to the dedicated upstream; dedicated upstreams carry their own timeout (2000 ms default) and circuit-breaker protection, falling back to the default engine on failure.

In addition, query concurrency is governed by **predictive capacity control** (Taylor-expansion remainders → a virtual-thread concurrency gate), modeled in 7.2.2 (ii); the dynamic-TTL model of the cache is in 7.2.2 (iii).

7.2 The Formulas

7.2.1 Definitions and Symbols

(i) Trend Prediction (upstream dynamic weights)

Symbol	Meaning	Default / range
rtt(t)	Round-trip time of the most recent successful query (milliseconds, nanoTime stamps; failures/timeouts record no sample)	measured
μ(t)	RTT EMA: μ(t)=α·rtt(t)+(1−α)·μ(t−1)	α=0.3
v(t)	First derivative (deterioration speed): v(t)=μ(t)−μ(t−1)	ms per sample
a(t)	Second derivative (acceleration): a(t)=μ(t)−2μ(t−1)+μ(t−2)	ms per sample²
σ(t)	Sample standard deviation of RTT (n−1 denominator)	window=5
I	Integrated health window: a boolean ring, recording once per second "not circuit-broken during this period"	N=10
J_i	Cost energy of node i (lower is better)	≥0
W_i	Published weight of node i (integer 0–100)	0–100
b	Integrated-health bonus (1.2 when the window is full and all-healthy, else 1)	1 / 1.2
current[i]	SWRR current-weight array	dynamic
weight[i]	Published weights used per SWRR round	0–100

(ii) Predictive capacity control (the query concurrency gate)

Symbol	Meaning	Default / range
C(t)	Load = live virtual threads + queued tasks (backlog injected via backlogSupplier, currently a constant 0)	measured
EMA(t)	Exponential smoothing of C(t): EMA(t)=0.3·C(t)+0.7·EMA(t−1)	Δt=1s
v, a, jerk	First/second/third differences	—
predict(t+5)	Taylor forecast: EMA + v·5 + 0.5·a·25	5 s lookahead
σ	Standard deviation of the last N=5 samples (noise test)	—
M	Largest absolute jerk over the last N=5 samples, clamped to maxJerkLimit	16.0

R_bound	Lagrange-remainder bound (M/6)·125	—
safetyThreshold	safety = max(MAX_VIRTUAL_THREADS − R_bound, MIN_SAFETY_THRESHOLD)	5000 / 1000

(iii) Frequency-aware dynamic TTL (the default cache engine; `-Ddns.cache.engine=caffeine` falls back to Caffeine)

Symbol	Meaning	Default / range
f	Domain access frequency (exponentially decayed): f ← f·exp(−λ·Δt)+1	λ=ln2/300 (half-life 300 s)
k	Half-saturation point of mult(f)	100
mult(f)	TTL multiplier: 2/3+(1.75−2/3)·f/(f+k)	[2/3, 1.75]
ttlBase	Minimum TTL recorded in the answer	—
s	Eviction access score: s ← 0.99·s+1 (steady-state cap ≈100)	—

7.2.2 Physical Meaning and Derivation

(i) The cost-energy function J (Trend Prediction)

```
J = exp(0.01·μ) + 2·max(0, v) + 1.5·|a| + σ/μ
```

Term	Penalizes	Physical meaning
exp(0.01·μ)	Absolute RTT level	Exponential penalty: every extra 100 ms of μ multiplies the term by ≈e, demoting high-latency nodes quickly
2·max(0, v)	Upward trend	Punishes only "getting slower" (v>0); getting faster earns nothing (guards against low-baseline gaming)
1.5·|a|	Curvature / jitter	The sharper the deterioration acceleration (spiking or oscillating), the higher the cost
σ/μ	Relative dispersion	The coefficient of variation — a scale-free stability measure

Cold-start convention: with no samples μ=σ=0 and σ/μ is mathematically undefined; the implementation defines that term's contribution as 0 (no observations, no dispersion) and leaves the other three intact.

Softmax weights:

```
logit_i = −0.5·J_i + ln(b_i)
W_i = exp(logit_i) / Σ_k exp(logit_k)
```

- A circuit-broken node gets W_i = 0 outright (BackoffController, a functional interface adapting UpstreamHealth.isOpen; node key is the host:port authority);
- The implementation exploits Softmax's shift invariance (subtract the max logit before exponentiating) to avoid exp(−0.5·J) underflowing to 0 and producing 0/0 at large J; the result is mathematically identical;
- The integrated-health bonus is equivalent to multiplying the unnormalized weight by b (adding ln(1.2) to the logit); while the window is unfilled or a breaker fires, b=1 — matching the formula verbatim.

Quantized publishing: W×100 is rounded into an AtomicIntegerArray (0–100); a healthy node whose rounded weight is 0 is lifted to 1 (anti-starvation); the soft cap means the total need not be 100 (SWRR does not require it). A single background daemon (dns-weight-scheduler) runs "sample → compute → publish" every 1 second and publishes once synchronously at startup; a failed round keeps the previous weights; the query path merely reads the array (lock-free, zero allocation).

SWRR (Smooth Weighted Round Robin):

```
each round: current[i] += weight[i] (all eligible nodes)
select:     the node i* with the largest current
deduct:     current[i*] −= Σ weight
```

- Conservation: two addAndGet calls per node keep the sum of all current strictly constant (net change 0 per invocation), so long-run selection converges exactly to the weight ratio; "pick max + deduct" is not one atomic operation — a deliberately accepted weakening of instantaneous smoothness (zero global locks on the query path);
- Nodes with weight ≤ 0 are excluded and have current zeroed (so a recovering node is not punished by its historical debt); if every weight is 0 the selector returns −1, the caller falls back to configuration order, and if all candidates remain broken the Forwarder degrades to UPSTREAM_FAILURE.

(ii) Predictive capacity control (Taylor remainders → the concurrency gate)

```
EMA(t)   = 0.3·C(t) + 0.7·EMA(t−1)                    (smoothing)
predict  = EMA(t) + v(t)·5 + 0.5·a(t)·25              (5-second Taylor forecast)
noise-lowering = if σ > 3.0, drop the acceleration term: predict = EMA + v·5   (Peano
remainder)
R_bound  = (M/6)·125, M = min(window max｜jerk｜, 16.0)  (Lagrange remainder)
safety   = max(5000 − R_bound, 1000)                  (safety threshold)
target   = clamp(ceil(predict), minPermits, floor(safety)), plus an anti-hunt dead-band of
1
```

- Peano remainder: when noise dominates (σ>3.0) the second-order term is untrustworthy, so the forecast drops to first order;
- Lagrange remainder: a bound M on the third derivative gives the error bound ｜R｜ ≤ M·h³/3! = M·125/6, which reserves the safety margin;
- Engineering layers: forecast path (single background thread, volatile publication, synchronized strictly forbidden) / resize path (low frequency) / reject path (fast-failing tryAcquire, lock-free);
- Semaphore hard constraints: growth goes through release(delta); shrinkage must use a private subclass exposing reducePermits (release cannot create an upper bound); a virtual thread is created only after tryAcquire succeeds, and permits are returned in finally;
- Coverage: UDP/DoT/DoH/DoQ admit per query (rejection semantics differ per protocol: UDP/DoT answer SERVFAIL, DoH answers HTTP 200 + DNS SERVFAIL, DoQ resets the stream per RFC 9250

§4.3); TCP/YiANLink are one-thread-per-connection and are excluded (connection-level governance is separate); rejection logging emits one WARN per 10,000 rejections (preventing log storms).

(iii) Frequency-aware dynamic TTL (the default cache engine)

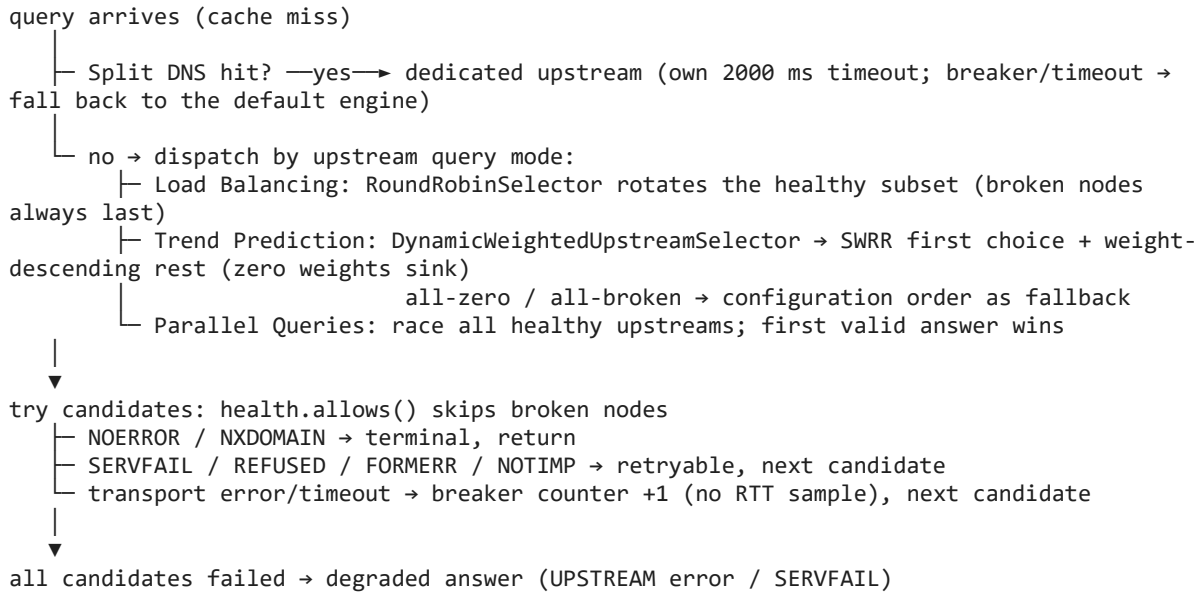
```
f ← f·exp(−λ·Δt) + 1      λ = ln2/300 (half-life 300 s)
mult(f) = 2/3 + (1.75 − 2/3)·f/(f + k)     k = 100 (Michaelis-Menten saturation)
ttlDyn = round(ttlBase · mult(f))
  → absolute safety clamp [10, 86400] seconds
  → user bounds: truncate by maxCacheTime first, then raise by minCacheTime (0 = unlimited)
```

- f=0 → mult=2/3 (cold domains shorten TTL); f=k → 1.2083; f→∞ → 1.75 (the hot-domain cap, avoiding stale-forever entries);
- Reads recompute f·exp(−λ·(now−lastAccess)); CAS retries take max(now, lastAccess) to survive clock setbacks;
- **Negative-cache divergence**: NXDOMAIN undergoes only the safety clamp and max truncation — it is never raised by min (the business meaning of a negative entry is "let the domain revive quickly");
- Capacity eviction: when full, 5 entries are sampled and the lowest decayed score s (s←0.99s+1) is evicted — no full sort;
- Transient errors (SERVFAIL/REFUSED/UPSTREAM-family) are never cached — the same semantics as the Caffeine fallback engine.

7.2.3 When Each Formula Runs

Formula	Trigger	Input	Output	Consumer
μ/v/a/σ update	Every successful upstream answer (network success counts, including deferrable rcodes)	rtt in ms	new Snapshot (CAS)	J computation
J + Softmax	Every second (scheduler thread) + once synchronously at startup	all node snapshots + breaker states	AtomicIntegerArray weights	SWRR
SWRR select	Every query needing an upstream	weight snapshot	first choice + weight-descending candidates	Forwarder
Split DNS match	After every cache miss	queried name	dedicated upstream or "no match"	Forwarder
Taylor forecast / gate resize	Every second	live virtual threads + backlog	latest permit target (dead-band suppresses ±1)	executeTask
Gate admission	Every query submission	current permits	pass / RejectedExecution	each transport (per-protocol rejection replies)
f/mult update	Every cache hit/store	Δt	new f, dynamic TTL	FrequencyAwareCache

7.3 Scheduling Decision Flow



Breaker state takes effect in real time (health.allows()), while weights publish once per second — for at most one second after a breaker fires, SWRR may still rank that node first; the Forwarder's health check skips it, so correctness is unaffected.

7.4 Tuning Guide

Parameter	Default	Adjustability	Effect
upstream.strategy	ROUND_ROBIN	hot via web	See 7.1 for the three modes; pick Trend Prediction under volatile hotspots, Parallel Queries when lowest latency matters
cache engine dns.cache.engine	frequency-aware (default)	JVM system property (read once at startup; switching requires a restart)	unset/frequency = frequency-aware; caffeine = W-TinyLFU fallback; unknown values log a warning and behave as frequency-aware
resetStateOnUpdate	false	hot via web	true resets breaker/RTT state whenever upstream config is saved (for troubleshooting); false preserves state
maxUpstreamAttempts	0 (all)	config/code	Limits parallel-race participants, controlling amplification
Split DNS timeoutMs	2000	per entry	Dedicated-upstream timeout; 0 = default
Gate maxVirtualThreads / minSafetyThreshold / maxJerkLimit	5000 / 1000 / 16.0	Gateway assembly constants	Raise for more concurrency; lower to reject earlier and protect small instances
α (EMA)	0.3	source constant	Larger reacts faster but is

			noisier
σ window / health window N	5 / 10	source constants	Bigger windows are smoother and recover more steadily
Health bonus b	1.2	source constant	Strength of the tilt toward consistently healthy nodes
Publish period	1 s	source constant	Shorter lowers breaker-awareness latency at some compute cost
f half-life / k / mult cap	300 s / 100 / 1.75	source constants	Set how fast hot-domain TTL grows and where it tops out
Cache TTL bounds	clamp[10,86400]; user min/max 0	config	minTtlSeconds/maxTtlSeconds govern the effective cache duration

Formula constants (α, windows, b, gate parameters) are built-in design values; changing them means changing source. Operations-level knobs are the configuration items above. Formula semantics must never be altered by engineering hardening.

7.5 Worked Examples

Example 1 — steady-state weights for three nodes

Let A/B/C hold steady at RTT = 30 / 60 / 120 ms (σ=0, v=0, a=0):

```
J_A = e^0.30 ≈ 1.350    J_B = e^0.60 ≈ 1.822    J_C = e^1.20 ≈ 3.320
logit = −0.5·J: −0.675 / −0.911 / −1.660
Softmax (normalize after subtracting the max logit): W_A ≈ 46.2%, W_B ≈ 36.5%, W_C ≈ 17.3%
quantized publish → 46 / 36 / 17 (sum 99; the soft cap allows it)
```

SWRR's long-run selection ratio is 46:36:17. When A then deteriorates (v>0), J_A rises and its weight falls automatically — "trend" awareness in action.

Example 2 — breaker isolation

B fails repeatedly past the threshold → isOpen → W_B=0 → current[B] zeroes in SWRR and B is never the first choice; after recovery, half-open probes succeed and the weight rebuilds over subsequent publish cycles (the health check skips B meanwhile — no bad answers leak).

Example 3 — predictive-gate arithmetic

C(t) series 100 → 110 → 120 (EMA seeded at 100):

```
EMA1 = 0.3·110 + 0.7·100 = 103
EMA2 = 0.3·120 + 0.7·103 = 108.1
v = 5.1, a = 108.1 − 2·103 + 100 = 2.1
predict(5s) = 108.1 + 5.1·5 + 0.5·2.1·25 = 159.85 → target permits 160
```

---

8. Configuration Reference

8.1 File Structure

Config files are YAML (config.yaml / config.yml) or JSON (config.json). The layered model is **built-in defaults → user file (deep merge) → runtime overlay**; unspecified fields fall back to defaults. Top-level blocks:

Block	Contents
server	DNS listeners and secure transport (bindAddress/ports/threads/hostname/tls/dot/doh/doq)
yianlink	The YiANLink proprietary protocol (enabled/port/psk)
upstreams	Upstream list (address + protocol)
upstream	Upstream query mode (strategy, resetStateOnUpdate)
domainRouting	Split DNS routing list
filter	Rule file paths and per-client grouping
cache	Cache switch/size/TTL
storage	Data-storage root
logging	Query log and runtime log level
web	The admin plane
plugins	Plugin-private settings (read via subset("plugins.<name>"))

Validation: illegal values aggregate into a ConfigError list printed one by one (exit 2); web saves run "validate everything → atomic write-back to the startup file → apply at runtime," and failed validation applies nothing. Change notifications compare per top-level block (no change, no notification).

8.2 Configuration Items

server (DNS listeners and secure transport)

Key	Type	Default	Description
server.bindAddress	string	0.0.0.0	DNS listening address
server.udpPort	int	53	UDP port (0 not allowed)
server.udpEnabled	bool	true	UDP switch (false skips UDP; TCP/DoT/DoH must cover at least one transport)
server.tcpPort	int	53	TCP port (0 = disable TCP)
server.threads	int	8	Worker threads
server.hostname	string	""	External service name (optional): validated against the certificate SAN/CN at startup; echoed in the banner

server.tls.certFile	string	TLS/fullchain.pem	PEM full chain (shared by DoT/DoH/DoQ; required when any is enabled)
server.tls.keyFile	string	TLS/privkey.pem	PEM private key (PKCS#8/PKCS#1; RSA only for DoQ)
server.tls.minVersion	string	TLSv1.2	DoH minimum TLS version (TLSv1.2/TLSv1.3); DoT is fixed TLS1.2; DoQ is fixed TLS1.3; TLSv1/1.1 are never supported
server.tls.legacyRsaCompat	bool	false	Legacy-client compatibility: also offers TLS_RSA_*; enabling needs a restart, disabling is immediate
server.dot.enabled / port	bool/int	false / 853	DoT server switch and port
server.doh.enabled / port / path	bool/int/string	false / 443 / /dns-query	DoH server; path must match ^/[a-zA-Z0-9_\-/]*$
server.doq.enabled / port	bool/int	false / 853	DoQ server; may coexist with DoT on the same port

yianlink (the YiANLink protocol)

Key	Type	Default	Description
yianlink.enabled	bool	false	Listens on TCP when enabled; needs a restart
yianlink.port	int	6833	Must not collide with plain TCP / DoT / DoH ports
yianlink.psk	string	""	256-bit Base64 pre-shared key (decodes to 32 bytes); required when enabled=true

upstreams / upstream (upstreams and query mode)

Key	Type	Default	Description
upstreams[].address	string	—	Accepts host / host:port / a full URL (e.g. `https://doh.pub/dns-query`, `tls://dot.pub`); DoH may carry a path; at least one entry
upstreams[].protocol	string	UDP	UDP / TCP / DOT / DOH (default ports 53/53/853/443); the address scheme must

			agree with the protocol, otherwise the entry is skipped with a warning
upstream.strategy	string	ROUND_ROBIN	ROUND_ROBIN load balancing / TREND_PREDICTION trend prediction / PARALLEL parallel queries; legacy values (SEQUENTIAL/RANDOM/PRIORITY/FAILOVER) normalize to ROUND_ROBIN
upstream.resetStateOnUpdate	bool	false	Whether saving upstream config resets breaker/RTT state (default: preserve)

Upstream address grammar (shared by upstreams and Split DNS via UpstreamAddressParser):

```
[scheme://][userinfo@]host[:port][/path]
- scheme only derives the default port and the DoH path; the protocol comes from protocol;
- scheme/protocol conflict → startup error; plaintext http:// DoH is rejected outright;
- a DoT host doubles as the TLS SNI; DoH keeps path and query;
- examples: https://doh.pub/dns-query (DOH), tls://dot.pub (DOT, same as dot.pub:853).
```

domainRouting (Split DNS)

Key	Type	Default	Description
domainRouting[].domainPattern	string	—	Exact name (www.a.com) or wildcard (*.b.com matches any-depth subdomains only, not b.com itself)
domainRouting[].upstreamAddress	string	—	[scheme://][userinfo@]host[:port][/path] (scheme=udp/tcp/dot/doh, default udp:53; shares UpstreamAddressParser with upstreams); pointing at the gateway's own listener is rejected at save time
domainRouting[].timeoutMs	int	2000	Dedicated-upstream timeout (0 = default 2000 ms); timeout answers SERVFAIL

filter (rules)

Key	Type	Default	Description
filter.blocklists[]	list	data/filters/ad-hosts.txt	Global blocklist files (double-bar Adblock blocks, hosts IP lines, bare domains, *. wildcards, /re/ regexes)
filter.allowlists[]	list	data/filters/allow.txt	Global allowlist (@@ exceptions beat blocklists)
filter.rewrites[]	list	data/filters/rewrites.txt	Rewrites (source domain + target IP/IPv6/CNAME, with automatic CNAME loop detection)
filter.clients[].client	string	—	Client IP/CIDR/comma mix or *
filter.clients[].blocklists[]	list	—	Group-specific lists

/ allowlists[]			(client-specific overrides global)

Rule syntax quick reference: `||domain^` Adblock block; `@@domain^` exception; `0.0.0.0 host` hosts block; `host target` rewrite line; `$important` high priority; `$client=1.2.3.0/24` per-client; `$dnstype=AAAA` per type; `/regex/` regular expression (with anti-catastrophic-backtracking guards); `|http://...|` URL anchoring.

cache

Key	Type	Default	Description
cache.enabled	bool	true	Cache switch (false = always go upstream)
cache.maxEntries	int	100000	Capacity cap (frequency-aware engine by default; W-TinyLFU capacity under the Caffeine fallback)
cache.defaultTtlSeconds	int	300	Default TTL for positive answers
cache.negativeTtlSeconds	int	60	NXDOMAIN negative-cache TTL
cache.minTtlSeconds	int	0	Minimum cache TTL (values below are raised; 0 = unlimited)
cache.maxTtlSeconds	int	0	Maximum cache TTL (values above are truncated; 0 = unlimited)

Cache engine: **frequency-aware dynamic TTL by default** (TTL floats with per-domain access heat — hot domains persist, cold ones expire quickly); the JVM flag `-Ddns.cache.engine=caffeine` explicitly falls back to Caffeine W-TinyLFU; unknown values log a warning and behave as frequency-aware. The engine is read once at startup — switching requires a restart; config fields and REST contracts are unchanged. Transient errors (SERVFAIL/REFUSED/UPSTREAM-family) are never cached.

storage

Key	Type	Default	Description
storage.dataDir	string	data	Base for query-log persistence; relative paths resolve against the jar directory; absolute paths used as-is

logging

Key	Type	Default	Description
logging.queryLogEnabled	bool	true	Query-log switch
logging.queryLogPath	string	querylog	Sub-directory relative to storage.dataDir (absolute paths

			used directly)
logging.retentionHours	int	168	Query-log retention in hours: 24/168/720/2160 or any custom value
logging.statsRetentionHours	int	24	Statistics window in hours: 24 = last day / 168 = last week
logging.maxEntries	int	100000	Ring-buffer capacity
logging.bufferSize	int	8192	Async write-queue capacity
logging.storageType	string	JSONL	JSONL (daily-rotated files) / MEMORY (memory only)
logging.level	string	INFO	TRACE/DEBUG/INFO/WARN/ERROR

web (admin plane)

Key	Type	Default	Description
web.enabled	bool	true	Admin-plane switch
web.bindAddress	string	127.0.0.1	Admin listening address (separate from the DNS data plane; intranet/localhost recommended)
web.port	int	8080	Admin port
web.adminUsername	string	admin	Administrator account
web.adminPassword	string	admin	Administrator password (⚠ change in production)

plugins (plugin-private settings): arbitrary key-value pairs read by plugins through DnsConfigProvider.subset("plugins.<name>"); example:

```
plugins:
  rate-limiter:
    qps: 5000
```

8.3 Typical Configurations

Scenario 1 — home ad blocking (minimal)

```
server: { bindAddress: 0.0.0.0, udpPort: 53, tcpPort: 53 }
upstreams:
  - { address: 223.5.5.5, protocol: UDP }
  - { address: 119.29.29.29, protocol: UDP }
filter:
  blocklists: [data/filters/ad-hosts.txt]
  clients:
    - client: 192.168.2.0/24          # children's devices
      blocklists: [data/filters/kid-block.txt]
cache: { enabled: true, maxEntries: 100000 }
web: { enabled: true, bindAddress: 127.0.0.1, port: 8080 }
```

Scenario 2 — public encrypted DNS (DoT + DoH + DoQ)

```
server:
  bindAddress: 0.0.0.0
  udpPort: 53
  tcpPort: 53
  hostname: dns.example.com
```

```
    tls:
      certFile: TLS/fullchain.pem
      keyFile: TLS/privkey.pem
    dot: { enabled: true, port: 853 }
    doh: { enabled: true, port: 443, path: /dns-query }
    doq: { enabled: true, port: 853 }
  web:
    enabled: true
    bindAddress: 10.0.0.5          # admin reachable from the intranet only
    port: 8080
```

Scenario 3 — a dedicated YiANLink channel (for in-house clients)

```
  yianlink:
    enabled: true
    port: 6833
    psk: "3q2+7wEYxkBeK0vF0uYn2O2o0S6mJ0yQ1aB9cD8eF0g="   # example: Base64 of 32 bytes
  server: { bindAddress: 0.0.0.0, udpPort: 53, tcpPort: 0 }
```

Scenario 4 — multiple upstreams (incl. URL syntax) + Trend Prediction + Split DNS

```
  upstreams:
    - { address: 223.5.5.5, protocol: UDP }
    - { address: "https://doh.pub/dns-query", protocol: DOH }
    - { address: "tls://dot.pub", protocol: DOT }
  upstream:
    strategy: TREND_PREDICTION
  domainRouting:
    - { domainPattern: "*.corp.local", upstreamAddress: udp://10.0.0.53:53 }
    - { domainPattern: "stream-vip.example", upstreamAddress: doh://dns.alidns.com, timeoutMs:
  0 }
---
```

9. Operations and Monitoring

9.1 Logging and Diagnostics

Query log (the logging block):

- Storage: daily-rotated JSONL `querylog-yyyyMMdd.jsonl` (under querylog/ within storage.dataDir), flushed in batches by an async virtual-thread writer; files older than retentionHours are cleaned at startup; an unwritable directory degrades to memory mode (WARN);
- Memory side: ring buffer (maxEntries) + bounded write queue (bufferSize; overflow increments droppedByBuffer) + 24-hour stat buckets (for /api/stats);
- Each record carries: timestamp, domain, query type, rcode, client IP, upstream, latency, source protocol (UDP/TCP/DOT/DOH/DOQ/YIANLINK), blocked flag, matched rule, cache-hit flag.

Diagnostics:

```
# grep today's log for a domain
grep "example.com" data/querylog/querylog-$(date +%Y%m%d).jsonl
# quick REST check
curl -H "Authorization: Bearer <token>" "http://127.0.0.1:8080/api/queries?limit=20"
```

Runtime log: stdout (level logging.level); add `-Ddns.tls.debug=true` for TLS handshake diagnostics including SNI.

9.2 Metrics

Upstream node metrics (dashboard / upstream servers page):

- `queries`: attempts — incremented before every attempt, including retries;
- `failures`: failed attempts (transport errors/timeouts/retryable rcodes);
- `avgLatencyMs`: recorded only after successful answers (failures are **deliberately not recorded**, to avoid polluting the scheduler's EMA); with no successful samples it returns **-1** (shown as "no successful answers"); **0** means a sample existed but took under 1 ms (shown as "<1 ms");
- Reading the numbers: "counts but no latency" = every attempt to that upstream was masked by failover (it never succeeded) — check reachability first.

Query metrics (GET /api/metrics):

Metric	Meaning
dns.query.total	Total queries
dns.query.dropped	Requests dropped by the pipeline
dns.query.filter.respond	Short-circuit answers by filters
dns.query.filter.allow / dns.query.blocked	Allowed / blocked
dns.query.rewritten	Rewrites (non-NXDOMAIN blocks)
dns.query.cache.hit	Cache hits
dns.query.latency	Average query latency
dns.protocol{UDP/TCP/DOT/DOH/DOQ/YIANLINK}	Counts per source protocol (a new protocol must sync the whitelist/colors/regression assertions)

Connection metrics (GET /api/connections / /api/status): active connections totalActiveConnections = TCP + DoT + DoH + yianLinkConnections; YiANLink relies on the 60s read timeout to reclaim half-open connections and zero its count.

System status (GET /api/system/info; pure JDK/JMX, zero third-party libraries):

Field	Semantics
uptime	JVM uptime; shown in hours below a day, days+hours above
load1/5/15	load1 from getSystemLoadAverage (unavailable on Windows → -1 → N/A); load5/15 are EMAs of 3-second samples (windows 300s/900s)
Threads	Current JVM thread count
Heap/stack/direct	Heap measured; stack = threads × per-thread stack size (reads -Xss, default 1 MB; an estimate); direct from the direct BufferPool
GC	YoungGC/OldGC counts and cumulative durations; recentGcPauseMs is the increment of cumulative GC time between samples (the pause of the last refresh window)
OS/Java	os.name/version/arch and java.version/vendor/runtime/vm (three monospace lines)

Unavailable values are uniformly -1 and rendered as N/A.

9.3 Troubleshooting FAQ

Symptom	Diagnosis and conclusion
nslookup always times out	First rule out local DNS hijacking/takeover tools (YogaDNS, ad blockers, security software) and port-53 squatting; bisect with a non-53 port + a raw UDP probe; do not treat it as a server bug while the data plane is healthy
Immediate exit, code 3	Port in use: locate the occupier via netstat/Resource Monitor; change the port or stop the conflicting process
Immediate exit, code 2	Config validation failed: the console lists each field path and reason — fix as instructed
Startup failure "certificate does not cover the external hostname"	server.hostname mismatches the certificate SAN/CN; replace the certificate or fix the hostname (the error lists the domains actually covered)
DoT clients fail while DoQ works	The certificate is untrusted or the address is outside the SAN (DoQ tools often skip verification); also note DoT is fixed to TLS1.2 with ALPN disabled — expected
ALPN warning logs	The client's ALPN was not on the whitelist and was explicitly rejected (by design); confirm the client

	protocol (h2/HTTP1.1/doq)
Web config save did not apply	Saves are whole-file atomic validations: check the returned validation errors; a failed disk write also applies nothing
A deleted-then-re-added upstream is already broken	Upstream state is reused by identity (authority + ordinal); to clear it use the "reset upstream state" button or resetStateOnUpdate=true
YiANLink connection count never zeroes	Half-open connections send no FIN — normal; the 60s read timeout reclaims them (regression-tested)
Forgot the admin password	Edit web.adminPassword in the config file → restart; all old sessions are invalidated
Low cache hit ratio	Check min/max TTL settings and the negative-cache TTL; transient errors (SERVFAIL etc.) are not cached by design; with many hot domains keep the default frequency-aware engine — to fall back, add -Ddns.cache.engine=caffeine
Load shows N/A on Windows	getSystemLoadAverage platform limitation (returns -1) — expected
Upstream "counts but no latency"	The upstream never answered successfully (all masked by failover): check the address/protocol/reachability; note DoH upstream addresses must be full URLs (e.g. https://doh.pub/dns-query) and the scheme must match the protocol

9.4 Upgrade and Rollback

Upgrade:

1. Back up: the config file (config.yaml) and data/ (query logs and rules);

2. Replace the jar (configs are backward compatible: legacy strategy values normalize; deprecated fields such as upstream weight are ignored and disappear from the list on the next save);

3. Start and check the banner version (including the codename);

4. Watch the dashboard and /api/system/info to confirm health.

Rollback: swap back the old jar + old config and restart; web saves write back to the config file, so restore from backup if needed.

Versioning: pom/manifest/jar filenames use the plain semantic version (e.g. 1.0.3); presentation surfaces (README, startup banner) append the codename "Marhaba" within the 1.0.0 → 1.2.20 range.

---

10. Developer Guide

10.1 Repository Layout

```
DNS-Program Files/
├── pom.xml                  # parent POM (aggregation + dependencyManagement)
├── build.ps1 / smoke.ps1    # Windows build/smoke scripts (with Maven environment fixes)
├── maven-settings.xml       # project-level Maven settings (mirror + local repository)
├── dns-server-api/          # interface layer
├── dns-core/                # dnsjava codec + answer factory
├── plugin-api/              # extension points (7 interfaces + sealed decisions)
├── config/                  # config model/loading/validation/hot-reload + examples
├── logging/                 # query log (JSONL/memory, stat buckets)
├── metrics/                 # in-memory metrics registry
├── filter-engine/           # filtering (rule parsing/indexes/client profiles)
├── upstream-resolver/       # transports/selectors/breakers/trend prediction/Split
DNS/address parsing
├── cache/                   # frequency-aware cache (default) + Caffeine fallback
├── dns-proxy/               # transport servers (UDP/TCP/DoT/DoH/DoQ/YiANLink), pipeline,
predictive gate, entry point
├── web-admin/               # admin REST API (Java) + frontend/ (Vue3 console)
├── tools/                   # smoke and other tools
└── config/                  # config.example.yaml / config.example.json
```

Key entry: `com.example.dns.proxy.DnsGatewayApplication` (main + 12-step assembly + graceful shutdown); front-end sources live in `web-admin/frontend/`, built into `web-admin/src/main/resources/static` (baked into the jar).

10.2 Development Environment

Step	Notes
1. JDK 21	Install and set JAVA_HOME (OpenJDK/Semeru/BiSheng all fine)
2. Maven 3.6+	Global mvn; this project always builds with `maven-settings.xml` (never rely on global settings)
3. Node.js (only for front-end work)	v20+; run `npm install` inside web-admin/frontend
4. IDE	Any; language level 21

Restricted environments: build.ps1 injects `MAVEN_OPTS=-Daether.syncContext.named.factory=noop` (works around file-lock interception; harmless elsewhere); the local repository path is set by maven-settings.xml.

10.3 Build, Test, Release

Task	Command
Full build (all unit tests + fat jar)	`powershell -File build.ps1 clean package`
Single-module tests (-am is mandatory)	`powershell -File build.ps1 -pl <module> -am clean test`
End-to-end smoke (package first)	`powershell -File smoke.ps1` (9 assertions)
Front-end build	`cd web-admin/frontend && npm run build` (output into resources/static)
Plain Maven (cross-platform)	`mvn -s maven-settings.xml -B clean package`

Fat-jar mechanics (maven-shade-plugin in dns-proxy): merges all internal modules and third-party dependencies; ServicesResourceTransformer merges SPI files; signature files and module-info are stripped; the manifest carries `Main-Class: com.example.dns.proxy.DnsGatewayApplication`, `Implementation-Version` and `Multi-Release: true` (dnsjava's multi-release jar requires excluding its InetAddressResolverProvider SPI so it never hijacks JDK-wide resolution). After packaging, the shade intermediate `original-*.jar` is removed so exactly one artifact remains.

Test scale: 350+ unit tests across the project; rough per-module magnitudes — dns-proxy 80+, config 60+, upstream-resolver 50+, filter-engine 60, web-admin 45, cache 30+.

Version bump procedure: replace the version in all 15 places (12 poms + frontend/package.json + package-lock.json + README) → grep for leftovers → `clean package` green → delete original-*.jar → verify via manifest and a real banner run. Codename rule: releases in the 1.0.0 → 1.2.20 range use "Marhaba"; beyond it a new codename must be confirmed.

10.4 Contribution Guidelines

- **Code baseline**: Java 21 (records/sealed/virtual threads idiomatic); no comments unless requested; names and variables in English, with user-facing strategies/features named semantically in Chinese and mapped to English constants (e.g. 趋势预测 → TREND_PREDICTION);
- **Architectural red lines** (violations count as regressions):

1. Keep the query hot path lock-free (lock-free published arrays / volatile snapshots; no synchronized on hot paths);

2. Formula semantics are immutable (hardening must not change the math; new rounding points must account for the anti-hunt dead-band);

3. The concurrency gate must "tryAcquire first, create the thread only on success, release in finally"; Semaphore shrinkage only via the reducePermits subclass path;

4. YiANLink must propagate the real peer clientIp (IP allow/block lists, rate limiting and logging depend on it);

5. A new protocol must sync: the protocol-counter whitelist, front-end colors, active-connection statistics, regression assertions;

6. The "cacheable rcode" decision exists in two places (DnsHandlerPipeline and CachingDnsCacheProvider) and must stay identical;

- **Testing**: every new behavior/fix carries unit tests; round-robin fairness uses strict numeric assertions (e.g. 3 nodes with 1 broken over 600 picks: exactly 300/300); throwaway verification code is deleted after use;
- **PR flow**: fork → branch → full `clean package` green + smoke passing → submit (with build/test evidence) → link the issue describing motivation and impact.

---

11. Security Notes

11.1 Security Model and Threat Assumptions

Trust boundaries: LAN clients (semi-trusted), the admin plane (administrator only), upstream DNS (untrusted link, trusted content), the public exposure surface (when encrypted protocols are opened externally).

Threat assumptions and controls:

Threat	Controls
Plaintext DNS eavesdropping/hijacking	DoT/DoH/DoQ encrypted transport (TLS1.2+, modern ECDHE/AEAD cipher whitelist; SSLv3/TLSv1/1.1 permanently blocked); proprietary YiANLink AES-256-GCM + 60s timestamp window against replay
Unauthorized clients on encrypted ports	DoT/DoH TLS certificate authentication; YiANLink PSK implicit authentication (Tag failure = disconnect)
ALPN/protocol-confusion attacks	ALPN whitelist; non-matching negotiation is explicitly rejected with a first-occurrence warning (diagnosable)
Unauthorized admin access	PBKDF2 (120k iterations) password hashing + constant-time comparison; Bearer sessions 8h; binds 127.0.0.1 by default; sensitive config masked
Password side channels	MessageDigest.isEqual constant-time comparison
Resource exhaustion / floods	Predictive capacity gate: fast rejection when permits run out (reject before spawning threads; rejection logs rate-limited); upstream concurrency gates and total timeouts
Malicious/abusive domains	Filter engine (important→allow→rewrite→block evaluation order) + IP allow/block lists and rate limiting (real clientIp throughout)
DNS amplification abuse	Proxy semantics without public recursion (disable UDP on demand, restrict listening segments)
Config half-writes/tampering	Whole-file atomic validation + atomic write-back; failed validation applies nothing; Split DNS rejects self-referencing targets (anti self-loop)
Upstream poisoning	Cache accepts only NOERROR/NXDOMAIN; SERVFAIL/REFUSED-type transients are never cached; DNSSEC upstreams optional
Certificate/domain spoofing	server.hostname startup validation against SAN/CN; the server does not validate SNI (any SNI receives the current certificate — a deliberate compatibility trade-off for single-certificate deployments)

11.2 Known Limitations and Best Practices

Known limitations:

1. Default password admin/admin (plaintext in the example config) — public knowledge; must be changed;

2. The admin plane is a single-administrator model without multi-user/RBAC; REST auth is a single Bearer token;

3. DoT is fixed to pure TLS1.2 with ALPN disabled (a compatibility trade-off, giving up TLS1.3); legacyRsaCompat relaxes to TLS_RSA_* (only for devices that cannot be upgraded);

4. DoQ (Kwik) accepts RSA private keys only (PKCS#8); EC/Ed25519 are unsupported;

5. Self-signed certificates need explicit client trust; the SAN must cover the hostname/IP clients use;

6. The YiANLink PSK is the root of trust: leakage means rotating keys everywhere; the protocol has no version-negotiation field (v1);

7. Query logs contain plaintext domains (privacy-sensitive; mind directory access control and retention);

8. Stack memory is an estimate (threads × stack size); Windows has no system load (-1 → N/A).

Best practices:

- Change the default password; keep web.bindAddress on the intranet/localhost; front the admin plane with a reverse proxy + extra authentication before any public exposure;
- Use a proper CA certificate and configure server.hostname validation when opening encrypted protocols externally;
- Generate a dedicated high-entropy PSK for YiANLink in production and rotate it periodically (both ends synchronized);
- Firewall down to exactly the needed ports; watch breaker and gate rejection counters (metrics semantics in 9.2);
- Refresh remote rule sources regularly; set retentionHours per compliance requirements;
- Back up config and data before upgrades; rehearse rollback as part of change management.

Appendices

Appendix A. YiANLink Frame Quick Reference

Transport frame (TCP 6833, one per message)

Field	Offset	Size	Description
Length	0	4B	Big-endian unsigned; = 12 + N + 16 (everything after itself)
IV	4	12B	Per-frame random nonce (never reused)
Ciphertext	16	N	AES-256-GCM encrypted plaintext
Auth Tag	16+N	16B	GCM authentication tag

Frame plaintext

Field	Offset	Size	Description
Timestamp	0	8B	Big-endian Unix ms; receiver requires ≤ 60 s skew, otherwise silently drops
StreamId	8	4B	Big-endian; answers must reuse the original StreamId
Payload	12	M	The complete DNS message (wire format, no TCP length prefix)

Appendix B. Full Formula Derivations

B1. Exponential term exp(0.01·μ): cost must rise monotonically in RTT with increasing marginal penalty (super-linear punishment of high latency). The exponential family μ → e^(0.01μ) gives 1 at μ=0 and ×e per 100 ms; the 0.01 coefficient sets that scale.

B2. Trend term 2·max(0, v): v>0 means "getting slower" and must be punished; v<0 earns nothing — preventing nodes from gaming rewards by oscillating off a low RTT baseline. The coefficient 2 ranks the trend term above the acceleration term.

B3. Curvature term 1.5· | a | : the second difference a is acceleration; taking its absolute value punishes both runaway deterioration and oscillation (alternating signs); 1.5 ranks it below the trend term.

B4. Dispersion term σ/μ (coefficient of variation): σ is absolute volatility; dividing by μ yields a scale-free relative measure; as μ→0 the expression degenerates, so the implementation defines the term as 0.

B5. Softmax shift-invariance proof: let m=max logit_k; then exp(logit_i − m)/Σexp(logit_k − m) = [exp(logit_i)·e^(−m)]/[e^(−m)·Σexp(logit_k)] = exp(logit_i)/Σexp(logit_k) — subtracting m changes nothing but prevents underflow.

B6. SWRR conservation proof: let S be the sum of current before a round. Adding weight to every node gives S+Σw; deducting Σw from the selected i* restores S. The sum therefore always equals the initial Σweight, and long-run selection converges strictly to the weight ratio.

B7. Taylor forecast and remainders: let the load curve f be three-times differentiable, $f(t+h)=f(t)+f'(t)h+f''(t)/2\cdot h^2+R$. With h=5: the first-order term is 5v and the second-order 12.5a. The Peano remainder $R=o(h^3)$ — under noise domination (σ>3.0) higher-order terms drown, so the forecast drops to first order. The Lagrange remainder $R=f^{(3)}(\xi)/3!\cdot h^3$ with $|f^{(3)}|\le M$ gives $|R| \le M\cdot 125/6\approx 20.83\cdot M$; M is estimated from the window's maximum jerk (clamped at 16.0), hence the safety margin R_bound=(M/6)·125, never below MIN_SAFETY_THRESHOLD=1000.

B8. EMA anti-hunting: in steady state a floating-point error such as EMA=6.000000000000011 ceils to 7 and the target would oscillate ±1 per second. A dead-band (no adjustment when |target−current| ≤1) absorbs the rounding tail; regression requires zero permit adjustments under steady sampling.

B9. Michaelis-Menten saturation mult(f): requirements — cold domains shrink TTL (mult(0)=2/3), hot domains grow it with a cap (mult(∞)=1.75), with a midpoint at the half-saturation k. The form mult(f)=A+(B−A)·f/(f+k) (standard saturation kinetics) gives A=2/3 at f=0, B=1.75 at f→∞, and (A+B)/2≈1.2083 at f=k.

B10. Frequency decay f←f·e^(−λΔt)+1: a first-order decay plus impulse injection; the half-life T½ satisfies e^(−λT½)=1/2 → λ=ln2/T½ with T½=300 s. Property: the steady state f* = 1/(1−e^(−λ)) for one access per second — heat converges instead of growing without bound.

Appendix C. CLI / API Quick Reference

CLI

```
# start (config probe: argument > ./config.yaml/yml/json > built-in defaults)
java -jar dns-proxy-1.0.3.jar [config-file-path]

# build / single-module tests / smoke
powershell -File build.ps1 clean package
powershell -File build.ps1 -pl dns-proxy -am clean test
powershell -File smoke.ps1

# verify DNS
nslookup baidu.com 127.0.0.1
kdig +tls @dns.example.com -p 853 example.com          # DoT
kdig +https @dns.example.com example.com               # DoH
kdig +quic @dns.example.com -p 853 example.com         # DoQ

# optional JVM flags
# (the frequency-aware cache engine is the default — no flag needed; these are
fallback/diagnostic)
-Ddns.cache.engine=caffeine    # fall back to the standard Caffeine W-TinyLFU engine
-Ddns.tls.debug=true           # TLS handshake diagnostics (with SNI)
```

REST API (auth: Authorization: Bearer <token>; login and static pages excepted)

```
POST /api/auth/login | POST /api/auth/logout | GET /api/auth/me
GET  /api/status | GET /api/connections | GET /api/metrics
GET  /api/queries?limit=N | GET /api/queries/detail | GET /api/stats?topN=N
GET/POST/PUT /api/rules?group= | GET/POST/PUT/DELETE /api/rules/sources | POST
/api/rules/sources/refresh
```

```
GET/PUT /api/config | GET /api/system/info
POST /api/system/restart | POST /api/system/shutdown
POST /api/maintenance/clear-query-log | POST /api/maintenance/clear-stats
POST /api/upstream/reset | GET /api/plugins
GET  /index.html (anonymous admin page)
```

Exit codes: 0 normal / 1 startup failure / 2 config error / 3 port in use.

Appendix D. Changelog

Version	Date	Major changes
0.1.0-SNAPSHOT	2026-10	12-reactor skeleton; plugin extension points (7 plugin-api interfaces)
YiAn1.0.0-RC1	2026-10-25 ~ 10-26	Phase 3 core (dnsjava codec/pipeline), Phase 4 upstream forwarding (four transports + breaker), Phase 5 caching (Caffeine), Phase 6 filter engine (AdGuard-style), Phase 7 layered config, Phase 8 query log (JSONL + stats), Phase 9 admin REST, Phase 10 entry point (12-step assembly/exit codes), fat-jar packaging
1.0.1_RC1	2026-10-27	Phase 11 secure transport: DoT/DoH/DoQ servers (Netty/Kwik, zero native), server.hostname validation
1.0.1_RC2	2026-10-28 ~ 10-30	Vue3 console front-end, split login auth, mock removal, DoH HTTP/1.1 support, explicit ALPN rejection, TLS hardening (full chain/session cache/handshake timeouts), DoT fixed TLS1.2, TLS_RSA compatibility switch, upstream query strategies (load balancing/parallel) with hot updates
1.0.2-RC1	2026-10-31 ~ 11-05	System status panel (/api/system/info), restart/shutdown, four-list rules page + remote sources, SNI/SslHandler hardening, YiANLink end-to-end (transport + front-end + monitoring contract), frequency-aware dynamic-TTL cache
1.0.3-RC2	2026-11-06 ~ 11-16	Predictive capacity control (Taylor-remainder gate) on the data plane;

		Trend Prediction dynamic-weight scheduling (J+Softmax+SWRR, three named strategies); manual weights retired; UpstreamStateRegistry hot-update state reuse; release codename Marhaba; Split DNS routing; upstream addresses accept full URLs (https://doh.pub/dns-query, tls://dot.pub) alongside bare names (UpstreamAddressParser, shared with Split DNS); CNAME rewrite chaining (answers attach the target's A/AAAA — no second query needed); upstream metric semantics fixed (queries = attempts, failures counter, latency recorded on success only, "no successful answers" / "<1 ms" display); the frequency-aware cache became the default engine; rules-page rewrite-table fixes and pagination focus retention; streaming rule loading (500k entries ≤ 150 MB)
1.0.3	2026-11-16 ~ 11-17	GA: full regression (all module tests + 9 smoke assertions) green; bilingual README/docs aligned (default cache engine, upstream URL grammar, metric semantics)

Per-version behavior details follow the README and the source; the build-level version is always the plain semantic version.

Appendix E. References

Category	Materials
DNS basics	RFC 1034 / RFC 1035 (DNS concepts and implementation), RFC 7766 (DNS over TCP)
Encrypted DNS	RFC 7858 (DoT), RFC 8484 (DoH), RFC 9250 (DoQ)
TLS/QUIC	RFC 8446 (TLS 1.3), RFC 5246 (TLS 1.2), RFC 9000/9001 (QUIC transport/TLS integration), RFC 6125 (certificate identity)
Algorithms	Nginx Smooth Weighted Round-Robin; Caffeine W-TinyLFU (Einbuch et al., 2017); Michaelis-Menten saturation kinetics; Taylor's theorem (Peano/Lagrange remainders)

Engineering	dnsjava 3.6.x docs; Netty 4.1 user guide; Kwik (pure-Java QUIC) docs; BouncyCastle PEM handling; Java 21 virtual threads (JEP 444)
License	Mulan PSL v2