Scalable Multi-Channel Notification System Design Documentation

1. Overview, Requirements, and Assumptions

A. Functional Requirements:

- Multi-Channel Delivery: Support email, SMS, and push notifications.
- Scheduled and Triggered Messaging: Allow both immediate and scheduled notifications.
- Personalization: Enable personalized message payloads per user.
- Retry and Fallback: Automatically retry failed notifications with backoff.
- Tracking and Logging: Record status (delivered, failed) for monitoring and analytics.

B. Nonfunctional Requirements:

- Low Latency: Process each notification in a few milliseconds.
- High Throughput & Scalability: Handle millions of notifications per day.
- Fault Tolerance & High Availability: Avoid single points of failure; use distributed, replicable services.
- Security: Use HTTPS for all communications; secure integrations with external providers.

C. Assumptions:

- The system must support high volume across channels (email, SMS, push) concurrently.
- External providers have their own rate limits and latencies.
- The design uses a distributed message broker, autoscaling workers, and a robust logging system.

2. High-Level Architecture and Component Responsibilities

A. API Gateway:

- Authenticates incoming notification requests.
- Routes requests via HTTPS to the Notification Engine.

B. Core Notification Service:

- Validates and enriches the notification payload.
- Schedules immediate or future notifications.
- Pushes jobs onto the Message Broker, partitioned by channel (email, SMS, push).

C. Message Broker:

- Acts as a queueing mechanism decoupling the ingestion of notifications from processing.
- Supports partitioning and high throughput.

D. Channel Workers:

- Email Worker: Processes email notifications by interfacing with SMTP providers or email APIs.
- SMS Worker: Handles SMS messages via external SMS gateways (e.g., Twilio).

- Push Worker: Manages push notifications via services like FCM and APNs.

E. Data Persistence and Analytics:

- A Notification Log Store maintains delivery status and history.
- A metrics system monitors performance and failure rates.

F. External Integrations:

- Email, SMS, and Push providers deliver the notifications.
- Fallback and alternative providers are used if the primary fails.

3. Detailed Workflow

A. Request Ingestion:

1. A client sends a notification request (with recipient ID, channel, and message content) via HTTPS to the API Gateway.
2. The API Gateway authenticates the request and routes it to the Notification Service.
3. The Notification Service validates and enriches the payload, then places a job on the Message Broker queue specific to the channel.

B. Worker Processing:

1. Workers for email, SMS, and push continuously poll the Message Broker for new jobs.
2. Each worker formats the message according to the external providers API requirements.
3. Workers send the notification to the external provider, handling response codes and errors.
4. Upon success or final failure, the worker logs the outcome in the Notification Log Store.

C. Monitoring and Feedback:

1. Metrics are gathered on processing latency, success rates, and provider performance.
2. Alerts are generated if failure rates exceed thresholds.

4. Scalability, Fault Tolerance, and Global Distribution

A. Scalability:

- API Gateway and Notification Service are stateless and horizontally scalable behind load balancers.
- Message Broker partitions jobs by channel, allowing independent scaling.
- Channel Workers auto-scale based on queue length and processing time.

B. Fault Tolerance:

- The message broker uses persistence and dead-letter queues for error handling.

- Data stores replicate logs for redundancy.
- Multiple instances of workers ensure that failure in one does not halt processing.

C. Global Distribution:

- Regional deployments and load balancers direct traffic to the nearest instance to reduce latency.
- External provider integrations may also be region-specific for optimal performance.

5. Protocols and External Infrastructure

A. Communication Protocols:

- All client-to-server communication uses HTTPS.
- Interservice communication uses gRPC or REST over secured TCP with mutual TLS.

B. External Providers:

- Email: Integrated with SMTP providers or services like SendGrid/Amazon SES.
- SMS: Connected to SMS gateways such as Twilio or Nexmo.
- Push: Uses Firebase Cloud Messaging (FCM) for Android and Apple Push Notification Service (APNs) for iOS.

C. Load Balancing and Autoscaling:

- Global and regional load balancers distribute API requests.
- Autoscaling policies adjust worker counts and service instances based on incoming load.

6. Final Thoughts

This design for a scalable notification system is structured to meet the demands of high throughput and low latency required for modern applications.

Key highlights include:

- Multi-channel support delivering notifications across email, SMS, and push.
- Decoupled architecture using a message broker and channel-specific workers, which allows independent scaling and robust error handling.
- Efficient and secure communications with external providers, with fallback mechanisms and detailed logging for observability.

The modular design ensures that the system can be extended further, for example by adding more channels or integrating additional external providers, while maintaining high performance and reliability globally.