Scalable Social Media Feed System Design Documentation

1. Overview, Requirements, and Assumptions

A. Functional Requirements:

- Enable users to create and share posts (text, images, videos).
- Aggregate posts from friends, groups, and followed pages into a personalized news feed.
- Provide real-time updates of new content, likes, and comments.
- Support reactions, comments, and sharing within the feed.
- Integrate content moderation and privacy controls.

B. Nonfunctional Requirements:

- Low Latency: Retrieve and update feeds within a few hundred milliseconds.
- High Throughput: Support billions of users and millions of posts per day.
- Scalability: Horizontally scale microservices, caching, and databases.
- Fault Tolerance: Ensure continuous service despite node or region failures.
- Security: TLS encryption, strong authentication, and fine-grained access control.

C. Assumptions:

- Billions of registered users; only a fraction are online simultaneously.
- The feed is generated using both push (fanout on write) and pull (on-demand computation) models.
- Social graph data is maintained in a dedicated service, and posts are stored in a distributed NoSQL database.

2. High-Level Architecture and Component Responsibilities

A. Client Tier:

- Mobile, web, and desktop apps interact via HTTPS for API calls and secure WebSockets for real-time updates.

B. Global Access and Routing:

- Global DNS directs users to the nearest region; regional load balancers distribute traffic to API Gateways.

C. API Gateway and Authentication:

- API Gateway authenticates, rate limits, and routes requests.
- Authentication Service issues secure tokens (JWT) to manage user sessions.

D. Social Graph and Content Ingestion:

- User Relationship Service maintains friend and follow networks.

- Content Ingestion Service collects posts from users and pushes them to the Feed Generation Service.

E. Feed Generation and Ranking:

- Feed Generation Service aggregates posts from user connections and shared pages.
- Ranking Engine orders posts based on recency, engagement, and personalization signals.
- An in-memory cache (e.g., Redis) stores precomputed feed data for low latency responses.

F. Data Persistence:

- Posts, user profiles, and social graph data are stored in distributed databases (NoSQL for posts, graph/relational for social data).

G. Monitoring and External Integrations:

- Global CDN, DNS routing, and analytics tools (Prometheus, Grafana) ensure low latency and high availability.

3. Detailed Workflow

A. Post Creation and Ingestion:

1. A user creates a post; the content and metadata are stored in the post service.
2. The post is pushed into the feeds of the user's friends and followers via a distributed message broker.

B. Feed Generation and Retrieval:

1. The Feed Generation Service aggregates relevant posts for a users feed.
2. The Ranking Engine scores posts based on various signals.
3. The final ranked feed is cached for quick retrieval via the API Gateway.

C. Real-Time Updates:

1. When a new post, like, or comment occurs, live updates are pushed to connected clients via WebSocket.
2. The cache is updated in real time to reflect the latest changes.

D. Personalization and Feedback:

1. User interactions and engagement metrics are recorded.
2. These metrics are fed back into the Ranking Engine to fine-tune personalized feed algorithms.

4. Scalability, Fault Tolerance, and Global Distribution

A. Horizontal Scalability:

- All microservices (API Gateway, Feed Generation, Social Graph) scale horizontally using containers/orchestration (e.g., Kubernetes).

- In-memory caching is partitioned and scaled (Redis cluster) to support millions of feed retrieval requests.

B. Fault Tolerance:

- Data stores are replicated; services are deployed in multiple instances with automatic failover.

- The system is designed for graceful degradation under partial outages.

C. Global Distribution:

- Multi-region deployments with global DNS routing and regional load balancers ensure that users experience low latency.

- CDNs cache static content and serve parts of the feed to reduce latency.

5. Protocols and Security

A. Communication Protocols:

- Client communications use HTTPS and secure WebSockets (WSS).

- Interservice communication is secured via gRPC or REST over TLS with mutual authentication.

B. Security:

- All data in transit is encrypted; sensitive data is encrypted at rest.

- Authentication is enforced using JWT tokens and RBAC controls.

- Regular audits and monitoring protect against unauthorized access.

6. Final Thoughts

This design for a social media feed system provides a robust, scalable framework for aggregating and personalizing content for billions of users. Key advantages include:

- A distributed, hybrid feed model that combines push-based delivery and on-demand computation for freshness and scalability.

- Advanced ranking algorithms that incorporate recency, engagement, and personalized user data.

- Global distribution with low latency via regional data centers, CDNs, and global DNS.

- High availability and fault tolerance through horizontal scaling, replication, and autoscaling mechanisms.

This architecture lays the foundation for a next-generation social media feed, similar to Facebooks, capable of delivering real-time, relevant content to billions of users worldwide.