



Evaluation of WhatsApp's Design




Evaluate the WhatsApp design and the associated non-functional requirements.

We'll cover the following

- Fulfill the requirements
- Trade-offs
 - The trade-off between consistency and availability
 - The trade-off between latency and security
- Summary

Fulfill the requirements

Our non-functional requirements for the proposed WhatsApp design are low latency, consistency, availability, and security. Let's discuss how we have achieved these requirements in our system:

- **Low latency:** We can minimize the latency of the system at various levels:
 - We can do this through geographically distributed WebSocket servers and the cache associated with them.
 - We can use Redis cache clusters on top of MySQL database cluster 
 - We can use CDNs for frequently sharing documents and media content. 
- **Consistency:** The system also provides high consistency in messages with the help of a FIFO messaging queue with strict ordering. However, 

the ordering of messages would require the **Sequencer** to provide ID with appropriate causality inference mechanisms to each message. For offline users, the Mnesia database stores messages in a queue. The messages are sent later in a sequence after the user goes online.

- **Availability:** The system can be made highly available if we have enough WebSocket servers and replicate data across multiple servers. When a user gets disconnected due to some fault in the WebSocket server, the session is re-created via a load balancer with a different server. Moreover, the messages are stored on the Mnesia cluster following the primary-secondary replication model, which provides high availability and durability.
- **Security:** The system also provides an end-to-end encryption mechanism that secures the chat between users.
- **Scalability:** Due to high-performance engineering, scalability might not be a significant issue, as WhatsApp can handle around 10 million connections per server. However, our proposed system is flexible, as more servers can be added or removed as the load increases or decreases.

Point to Ponder

Question

In the event of a network partition, what should the system choose to compromise between consistency and availability?

Show Answer ▼

Approaches to Achieve the Non-functional Requirements



Non-functional Requirements	Approaches
Minimizing latency	<ul style="list-style-type: none">• Geographically distributed cache management systems and CDNs
Consistency	<ul style="list-style-type: none">• Provide unique IDs to messages using Sequencer or other mechanisms• Use FIFO messaging queue with strict ordering
Availability	<ul style="list-style-type: none">• Provide multiple WebSocket servers and managers to establish connections between users• Replication of messages and data associated with users and groups across different servers• Follow disaster recovery protocols
Security	<ul style="list-style-type: none">• Via end-to-end encryption
Scalability	<ul style="list-style-type: none">• Performance tuning of servers• Horizontal scalability of services



Trade-offs

We've seen that our proposed WhatsApp system fulfills the functional and non-functional requirements. However, two major trade-offs exist in the proposed WhatsApp design:

- There's a trade-off between consistency and availability.
- There's a trade-off between latency and security.



The trade-off between consistency and availability



According to the CAP theorem, in the event of a network failure or partition, the system can provide either consistency or availability. So, in the case of our WhatsApp design, we have to choose either consistency or availability.

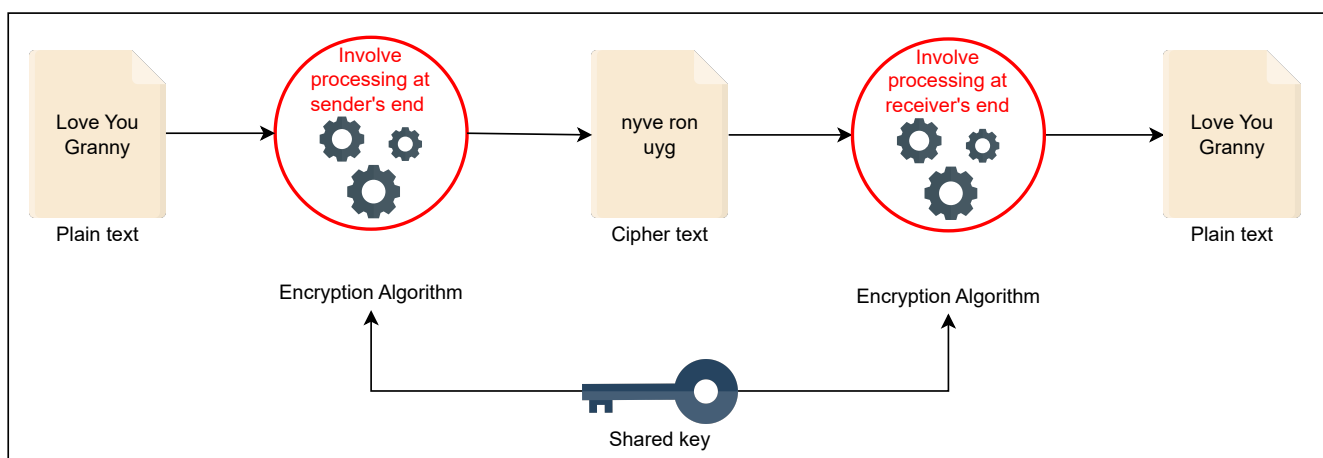


In WhatsApp, the order of messages sent or received by a user is essential. Therefore, we should prioritize consistency rather than availability.

The trade-off between latency and security

Low latency is an essential factor in system design that provides a real-time experience to users. However, on the other side, sharing information or data over WhatsApp might be insecure without encryption. The absence of a proper security mechanism makes the data vulnerable to unauthorized access. So, we can accept a trade-off prioritizing the secure transmission of messages over low latency.

We might wonder where the trade-off is. Often, communication involves multimedia. Encrypting them in near real-time on the sender device and decrypting on the receiver side can be taxing for the devices, causing latency. The process is illustrated in the following figure:



End-to-end encryption causes a delay in processing, which has an impact on latency

Summary

In this chapter, we designed a WhatsApp messenger. First, we identified the functional and non-functional requirements along with the resources estimation crucial for the design. Second, we focused on the high-level and detailed design of the WhatsApp system, where we described various



components responsible for different services. Finally, we evaluated the non-functional requirements and highlighted some trade-offs in the design.



This design problem highlighted that we can optimize general-purpose computational resources for specific use cases. WhatsApp optimized its software stack to handle a substantially large number of connections on the commodity servers.

[← Back](#)[Next →](#)[Detailed Design of WhatsApp](#)[Quiz on WhatsApp's Design](#)[Mark as Completed](#)