Overview of Communication Mechanisms Adopted in Modern Systems

Authors: Michał Żdanuk, Jakub Szczygielski

Supervisor: Radosław Roszczyk

Introduction

Modern software systems base on efficient communication mechanisms to ensure scalability, reliability, and performance. As systems become increasingly advanced, they often consist of numerous independent components that must exchange data seamlessly. Effective communication between these components is essential for maintaining system integrity and responsiveness. The aim of this research was to review key communication mechanisms used in modern systems. A multi-stage selection process was conducted to identify the most relevant scientific articles, summarize research findings, and determine which methods perform best in specific contexts.

Methodology

This research methodology is structured as a four-step process designed to systematically identify, evaluate, and analyze relevant scientific resources. The step-by-step approach ensures the selection of the most suitable papers and allows for deriving meaningful conclusions.

Step 1: Define search terms, select digital databases, and organize search results using Zotero for efficient management.

Step 2: Review materials by titles and abstracts, applying exclusion criteria like publication dates from 2015 onwards.

Step 3: Further extraction by evaluating introductions and results, focusing on methodologies and clear comparative data.

Step 4: Conduct an in-depth analysis of selected articles to derive conclusions about communication mechanisms in systems.

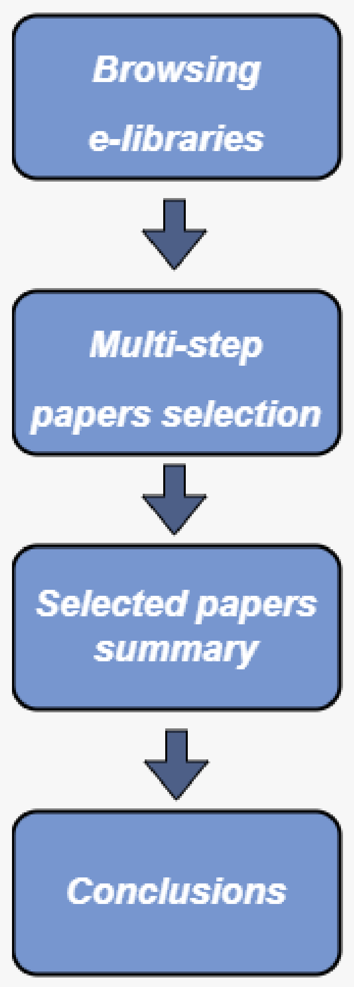

Figure 1: Full process of conducted research

Papers selection process

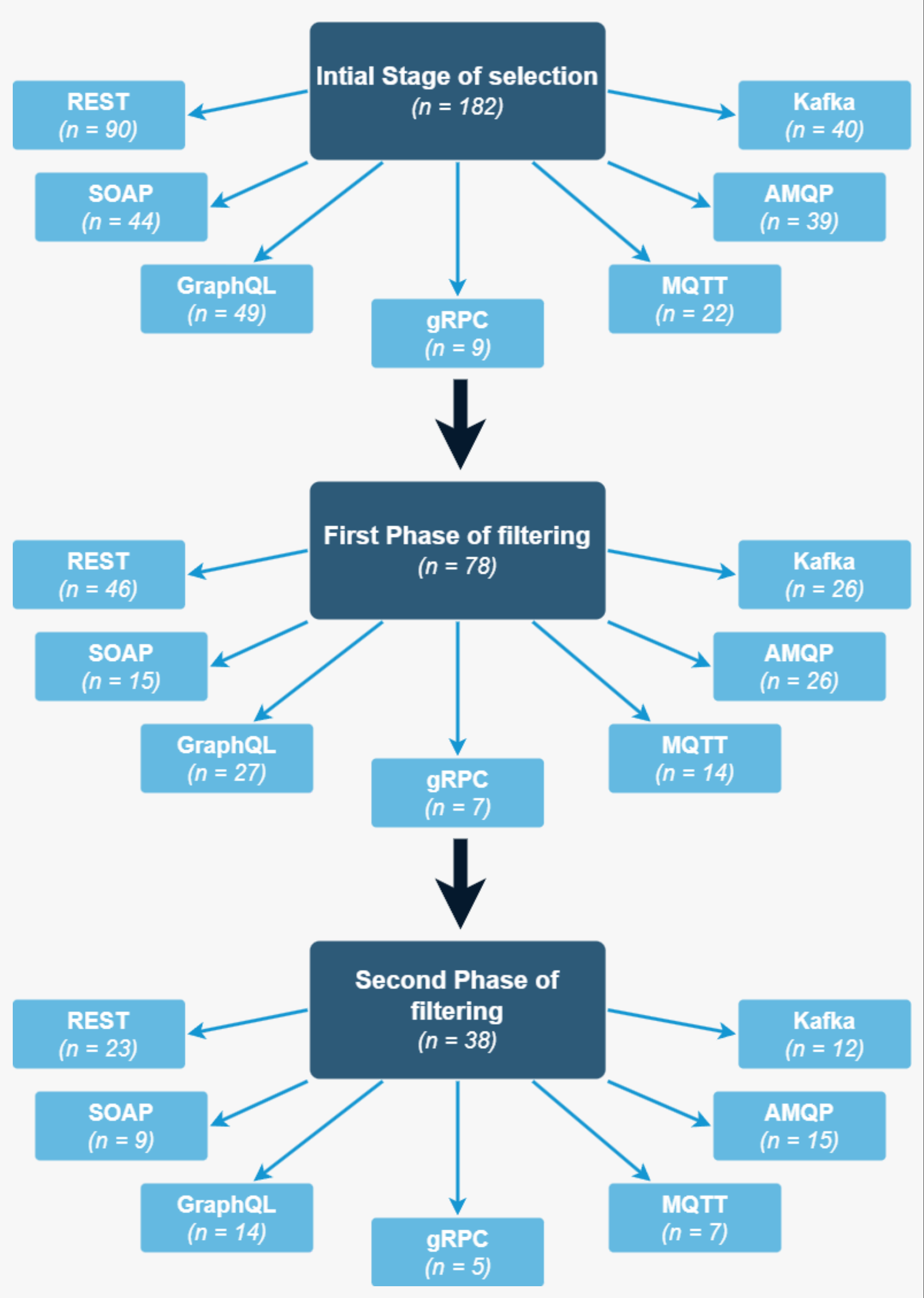

Figure 2: Multi-step process of papers selection

Final set of articels

No	Source	The number of sources
1	BazTech	9
2	Scopus	7
3	IEEE Xplore	11
4	Web of Science	4
5	Google Scholar	7

Table 1: Comparison of sources based on number of publications

Table 1 shows the final publication count from each digital library. IEEE Xplore contributed the most (11), followed by BazTech (9), while Scopus, Google Scholar, and Web of Science had 7, 7, and 4 publications, respectively. This reflects the stringent inclusion criteria ensuring data quality and relevance.

No	Communication mechanism	The number of sources
1	REST	23
2	SOAP	9
3	GraphQL	14
4	gRPC	5
5	Kafka	12
6	MQTT	7
7	AMQP	15

Table 2: Comparison of comunication mechanisms based on number of sources

Table VI shows the final count of sources on communication mechanisms: REST (23), AMQP (15), GraphQL (14), Kafka (12), SOAP (9), MQTT (7), and gRPC (5), highlighting the dominance of modern, scalable mechanisms like REST, GraphQL, and AMQP.

Results & Conclusions

Our review and research did not identify a single, universally ideal communication mechanism applicable to every scenario. The choice of a communication method should be dictated by the application's specific needs, the type of data being processed, and the requirements for performance, security, and scalability.

General Findings on Each Communication Method:

• REST remains the most widely studied and commonly used communication mechanism due to its simplicity and flexibility in web applications.

• GraphQL is efficient for precise data selection and retrieving large datasets, but it may be less effective for simple services with a limited number of resources. Studies indicate that GraphQL consumes more CPU resources and has longer response times, especially when dealing with nested data and multiple queries.

• gRPC offers the fastest data transfer, particularly in streaming scenarios and high-load environments. However, it is less frequently represented in research compared to other mechanisms.

• SOAP, while providing better security and reliability, comes with higher overhead and lower efficiency compared to REST. Research shows that REST outperforms SOAP in response time and energy consumption, particularly in mobile and cloud environments.

• Kafka is characterized by high throughput and is well suited for long-term message storage, making it an excellent choice for large-scale data streaming systems.

• MQTT is preferred in resource-constrained environments due to its lightweight nature, whereas AMQP offers greater reliability and enterprise-grade features, making it ideal for applications requiring high security and inter-operability.

Practical Guidelines for Developers

• For simple services with a small number of resources, REST is often the best choice, while for complex services with multiple resources, GraphQL can be more efficient.

• In systems with high throughput requirements, Kafka is the preferred option, whereas RabbitMQ may be better suited for systems requiring guaranteed message delivery and low latency.

• When designing IoT systems, MQTT is recommended for devices with limited resources, while AMQP is better suited for applications demanding reliable data transmission.