Background

ABC Ltd. is building an online store that uses the Microservice architecture pattern and are implementing the product details page. They need to develop multiple versions of the product details user interface:

* HTML5/JavaScript-based UI for desktop and mobile browsers - HTML is generated by a server-side web application.
* Native Android and iPhone clients - these clients interact with the server via REST APIs.

In addition, the online store must expose product details via a REST API for use by 3rd party applications.

A product details UI can display a lot of information about a product. For example, the Amazon.com details page for any Action displays:

* Basic information about the book such as title, author, price, etc.
* Your purchase history for the book
* Availability
* Buying options
* Other items that are frequently bought with this book.
* Other items bought by customers who bought this book.
* Customer reviews
* Sellers ranking

Since the online store uses the Microservice architecture pattern the product details data is spread over multiple services. For example,

* Product Info Service - basic information about the product such as title, author
* Pricing Service - product price
* Order service - purchase history for product
* Inventory service - product availability
* Review service - customer reviews etc.

Consequently, the code that displays the product details needs to fetch information from all these services.

Problem Context

Let us understand the problem of direct communication between the client app and microservices. Here, we have five services, and those services are directly interacting with the client applications. In this scenario, the client can call directly to any services which can create chaos, and operational and maintenance costs will be significantly higher across the system, as we are allowing any client directly to use services as per their wish. This problem grows even more chaotic when the number of clients increases, especially if third-party vendors start consuming APIs.

Secondly, we need to have public IP for each microservice, and those IPs will be exposed publicly through the internet. As a result, we are providing more surfaces for attackers to attack.

Also, implementation of cross-cutting concerns such as authorization, data transformations, and dynamic request dispatching b/w client app and services can require significant development effort on each service. This means we'll have lots of duplicated code and code rework is required in case any modification. That is not a good practice in general.

Most of the times client apps need to call multiple inter dependant services to implement certain functionality, which required multiple API calls. Multiple calls mean multiple network round trips between the client and the server which causes significant latency.

To overcome the various shortcomings of direct client-to-microservice communication identify the pattern who can reduce the number of requests that the application makes to backend services and improve application performance over high-latency networks.