IF3230 – Sistem Terdistribusi

Arsitektur

Achmad Imam Kistijantoro (imam@informatika.org)
Judhi Santoso (judhi@informatika.org)
Anggrahita Bayu Sasmita (bayu.anggrahita@informatika.org)

Maret 2022

Informatika – STEI – ITB

- *Architecture is about the important stuff…whatever that is.*
  - Ralph Johnson

To read: Fowler, Martin. "Who needs an architect?." IEEE SOFTWARE 20, no. 5 (2003): 11-13.
https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

- definisi IEEE:

the **highest level concept** of a system in its environment. The architecture of a software system (at a given point in time) is its **organization or structure** of **significant components** interacting through **interfaces**, those components being composed of successively smaller components and interfaces

Aplikasi Terdistribusi

- Terdiri atas multiple proses, yang berjalan pada satu atau lebih komputer/node
- Komunikasi antar proses dilakukan di atas infrastruktur tertentu
  - Socket/TCP/UDP, HTTP, WebSocket, etc.
- Mungkin pada lingkungan heterogen:
  - Mobile, linux, windows, Internet, LAN etc
  - Dikembangkan dengan bahasa/teknologi berbeda: e.g. python, C, Javascript

Hal yang perlu diperhatikan

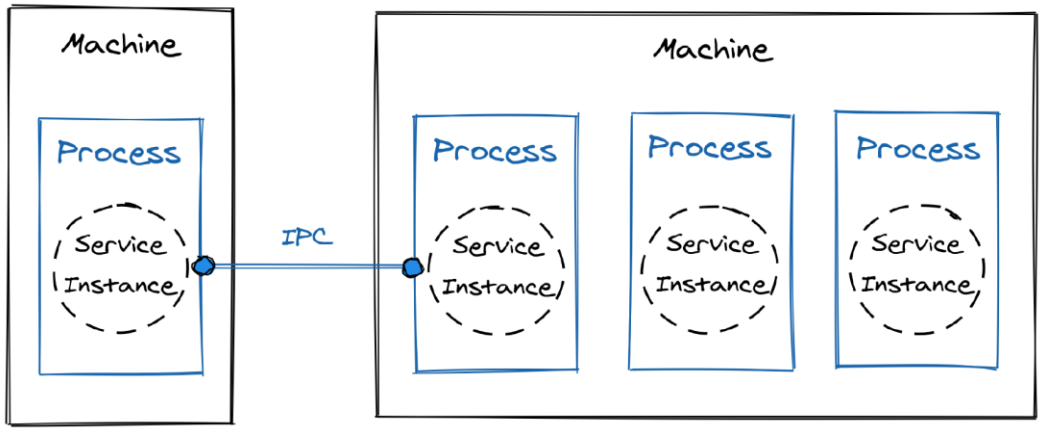
- Komunikasi
  - Mekanisme pengiriman data antar node
  - Bagaimana request dan response dikirimkan (e.g. via HTTP, TCP etc)
  - Menggunakan library/framework=> aware dengan abstraction leak (https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/). Perlu memahami stack komunikasi yang digunakan
- Koordinasi
  - Penanganan siapa melakukan apa dan bagaimana jika node jika gagal
- Skalabilitas
  - Efisiensi dalam menangani beban: throughput dan latency/response time
- Resiliensi
  - Mampu beroperasi meskipun terjadi kegagalan
- Operations
  - Proses pengelolaan, mulai dari development, testing, deployment, maintain

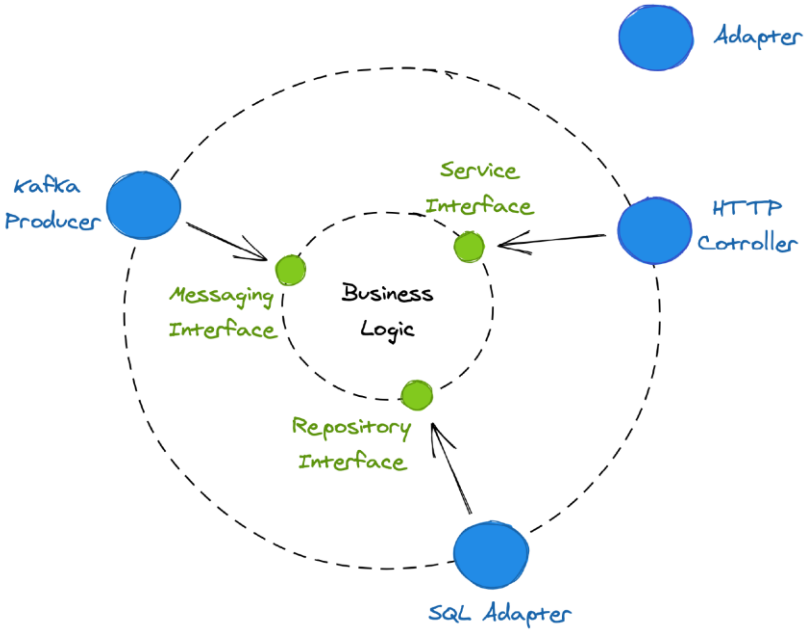
Anatomi sistem terdistribusi

- Umumnya, secara fisik, sistem terdistribusi adalah sekumpulan mesin yang terhubung melalui jaringan.
- Saat run-time, sistem terdistribusi terdiri atas sejumlah proses software yang berkomunikasi melalui mekanisme *interprocess communication* (IPC), seperti misalnya HTTP
- Dari perspektif implementasi, system terdistribusi terdiri atas sekumpulan *loosely-coupled components* yang dapat di-deploy dan di-scale secara independent yang disebut sebagai *services*.

Anatomi Sistem Terdistribusi

- Service: mengimplementasikan bagian tertentu dari keseluruhan kapabilitas system. Di dalam sebuah service terdapat business logic, yang menyediakan *interface* untuk berkomunikasi dengan komponen lain.
- Interface dapat berupa inbound (e.g. API), atau outbound (API call yg dilakukan oleh service tsb)

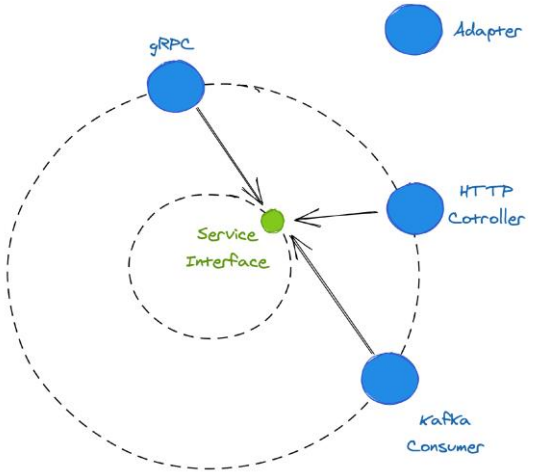




Vitillo, Understanding Distributed Systems,2021

API

- Sebuah service menyediakan layanan sebagai interface, dan diakses melalui adapter.
- Komunikasi dapat berupa direct, atau indirect melalui broker
- direct: client sends request, server replies response
- Reply-response message berisi data yang di-serialisasi yang language-agnostic.



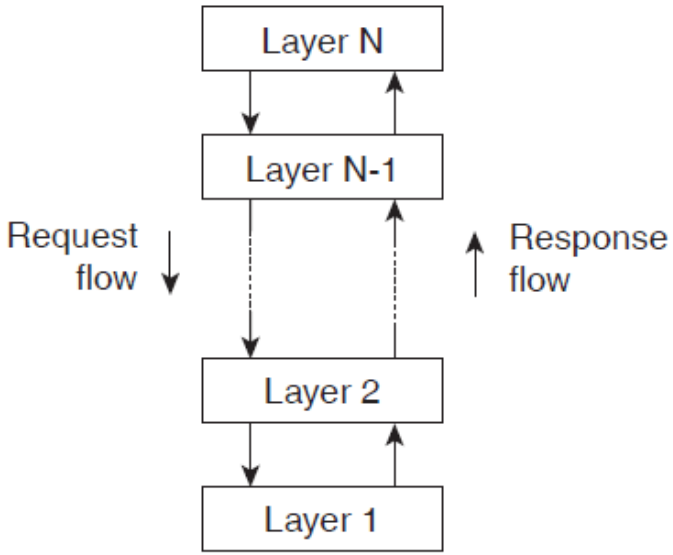
Vitillo, Understanding Distributed Systems,2021

API

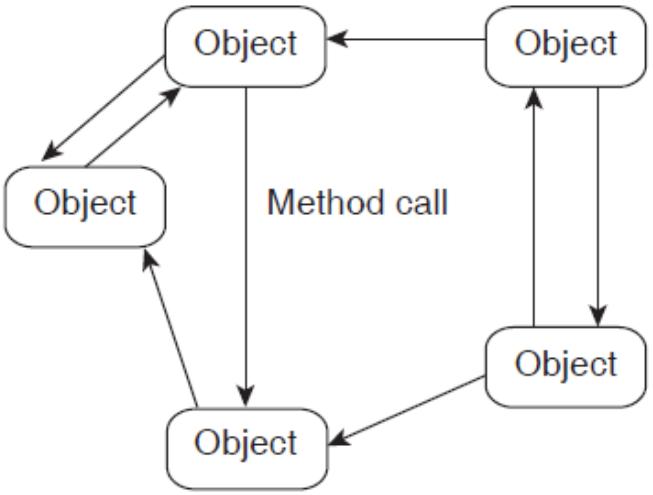
- Saat client mengirim request, dan terblok menunggu response: synchronous communication
  - Tidak efisien, karena memblok thread yang seharusnya bisa melakukan aktivitas lain
- Teknologi yang sering digunakan: gRPC, REST, GraphQL

Architecture Styles

- membagi tanggungjawab menjadi komponen-komponen, dan mendistribusikan pada mesin yang berbeda



(a)

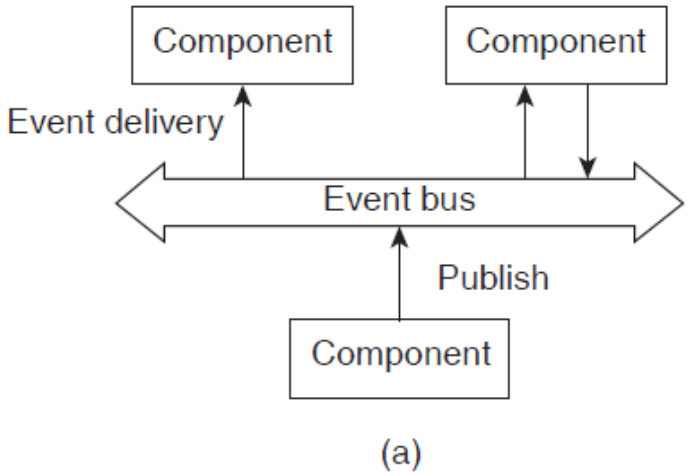


(b)

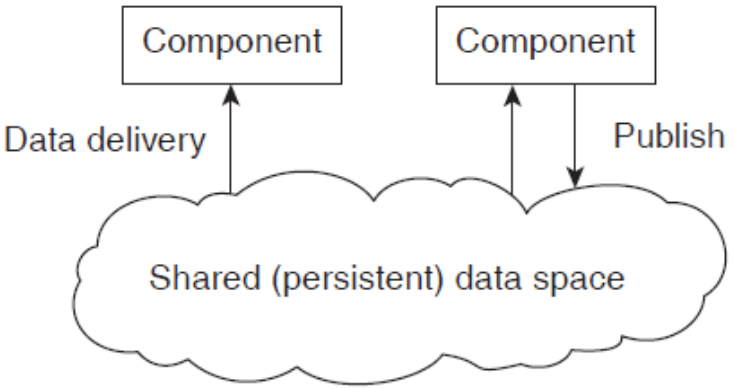
a) Layered style for client-server systems
b) Object-based style for distributed object systems

Architecture Styles

- Decoupling/memisahkan proses in space and time



(a)

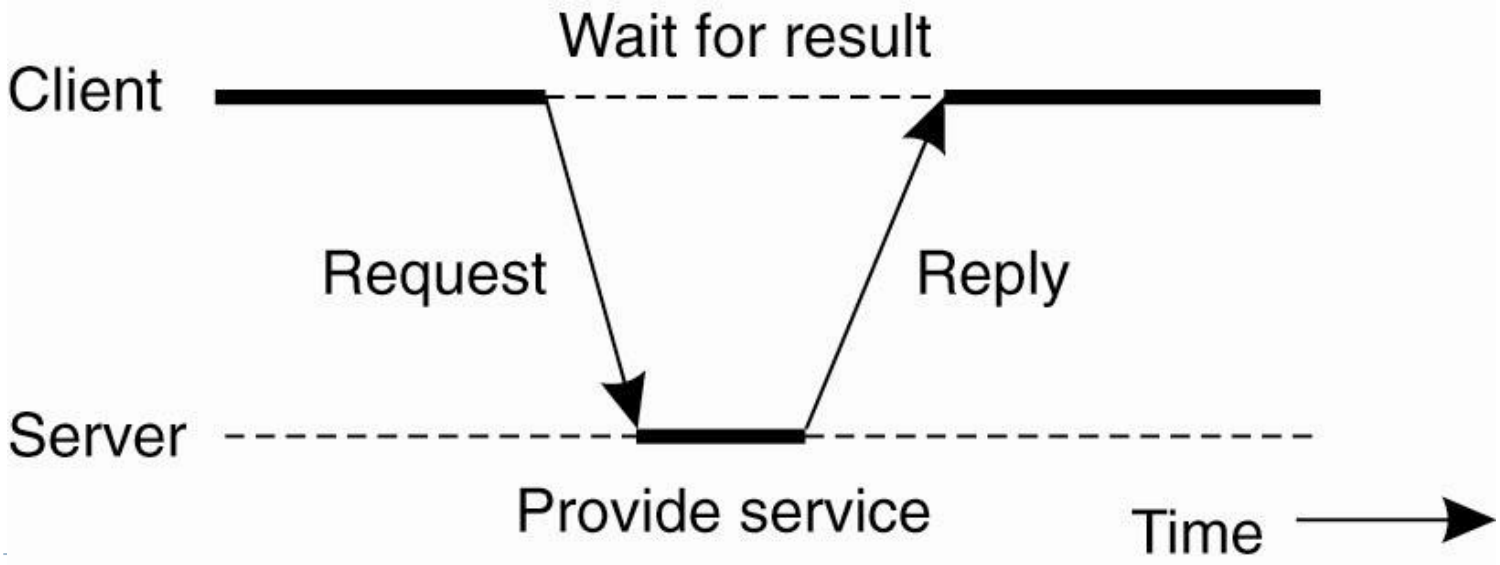


(b)

a) publish/subscribe (decouple in space)
b) shared data space (decouple in space and time)

Centralized Architecture

- Basic client server model
  - Proses yang memberikan layanan (servers)
  - proses yang menggunakan layanan (clients)
  - clients dan servers dapat berada pada mesin berbeda
  - follow request/reply model

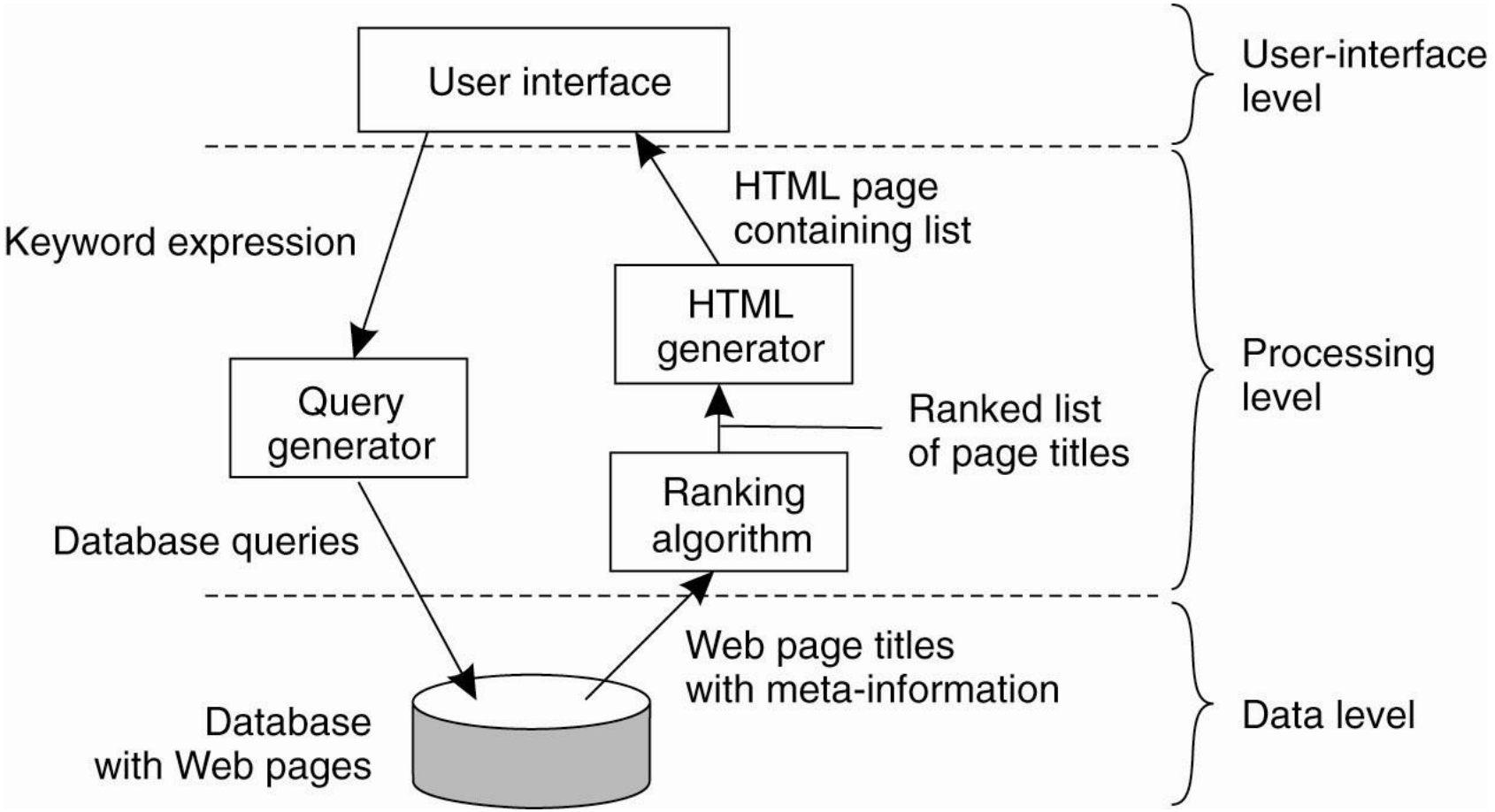


Application Layering

- Traditional three-layered view
  - UI layer
  - Processing layer
  - Data layer

- Model layer ini sering digunakan pada pada distributed information systems, menggunakan teknologi database dan aplikasi terkait

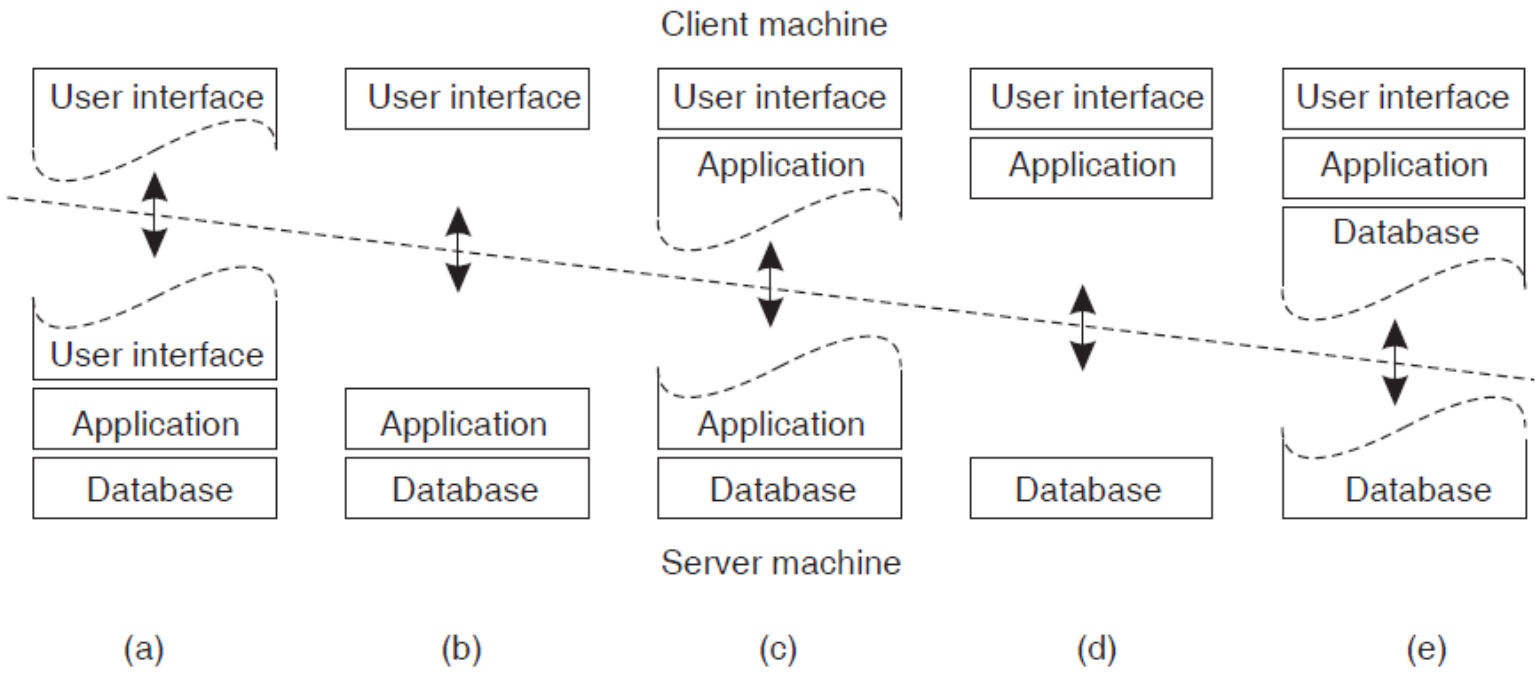
Application Layering



Multi-tiered architecture

Single-tiered: dumb terminal/mainframe configuration
Two-tiered: client/single server configuration
Three-tiered: each layer on separate machine

Traditional two-tiered configurations:



Service-based architecture

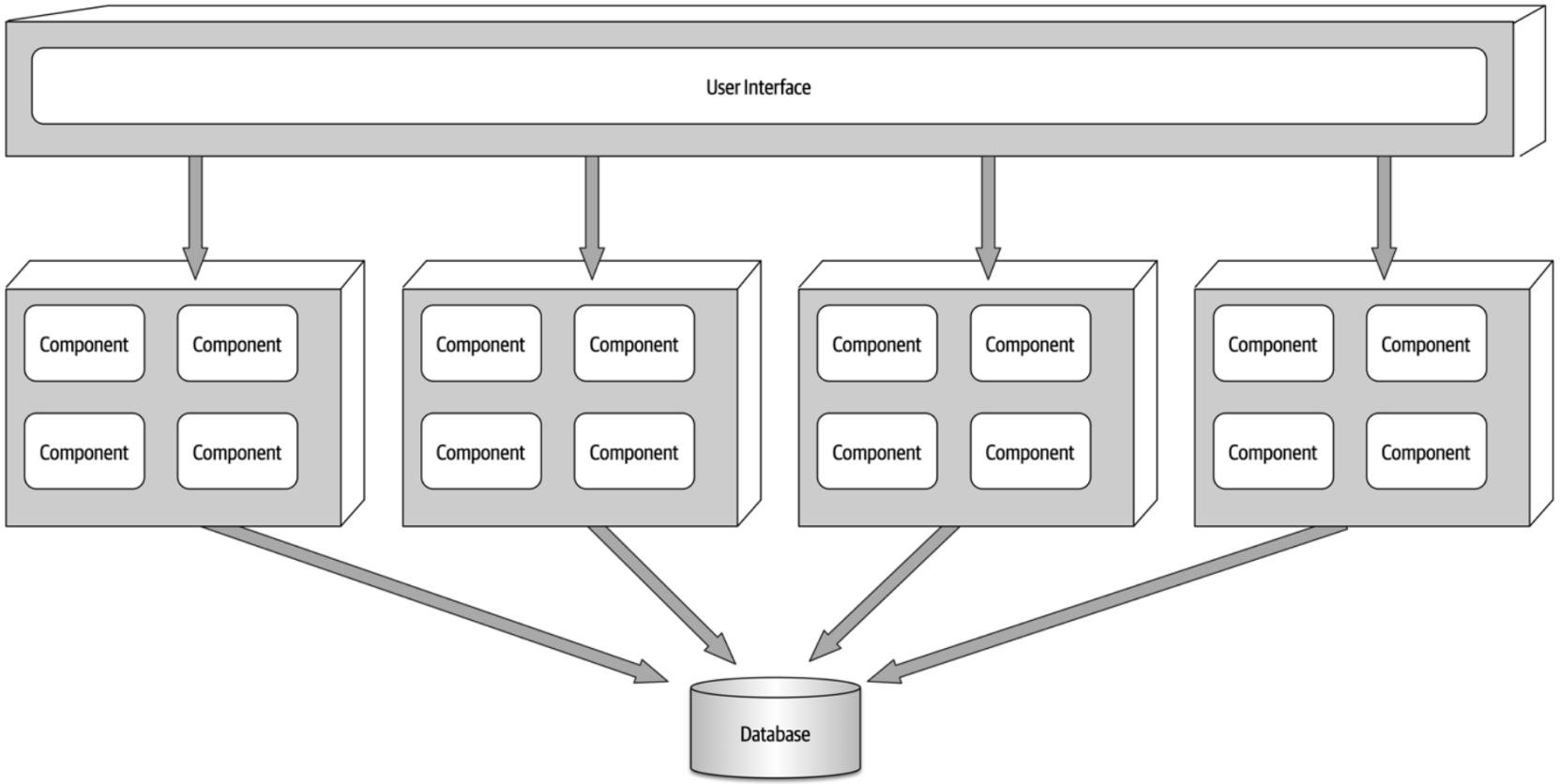


Figure 13-1. Basic topology of the service-based architecture style

Service based architecture

- Mengelompokan service berdasarkan domain
- Akses umumnya melalui REST, RPC (gRPC), SOAP
- Sering menggunakan shared database
- Sering menggunakan intermediate/API Gateway

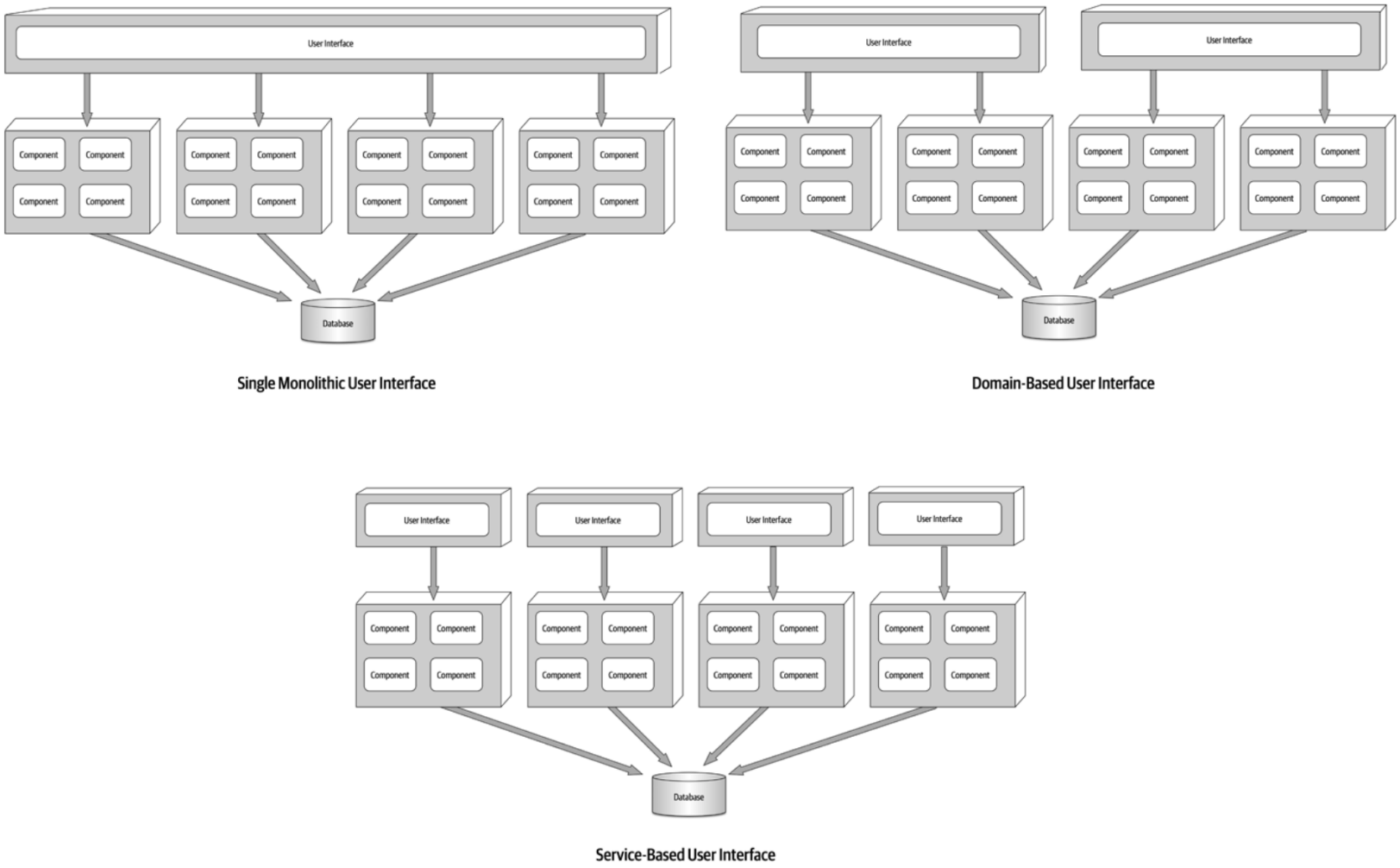


Figure 13-2. User interface variants

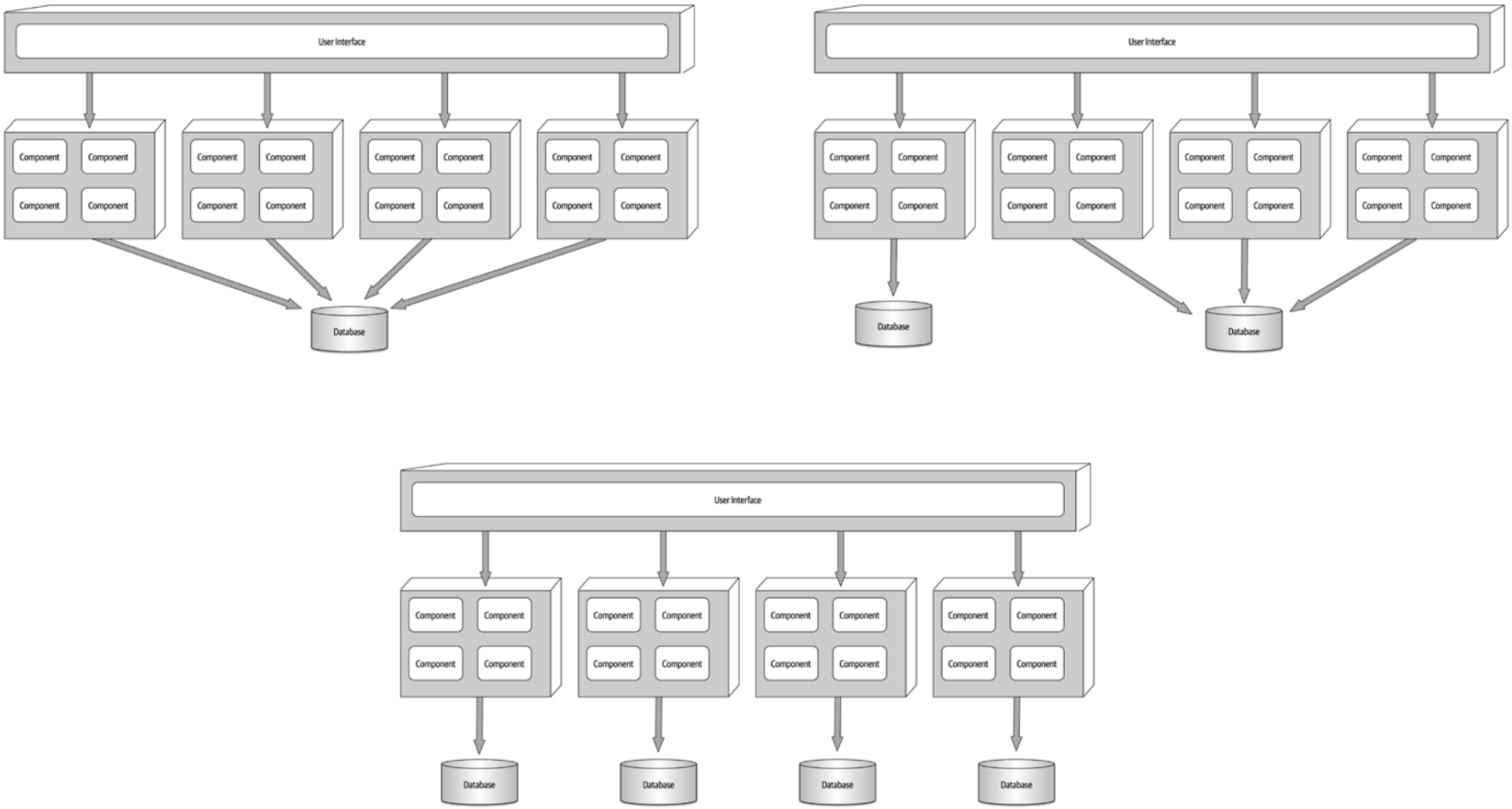


Figure 13-3. Database variants

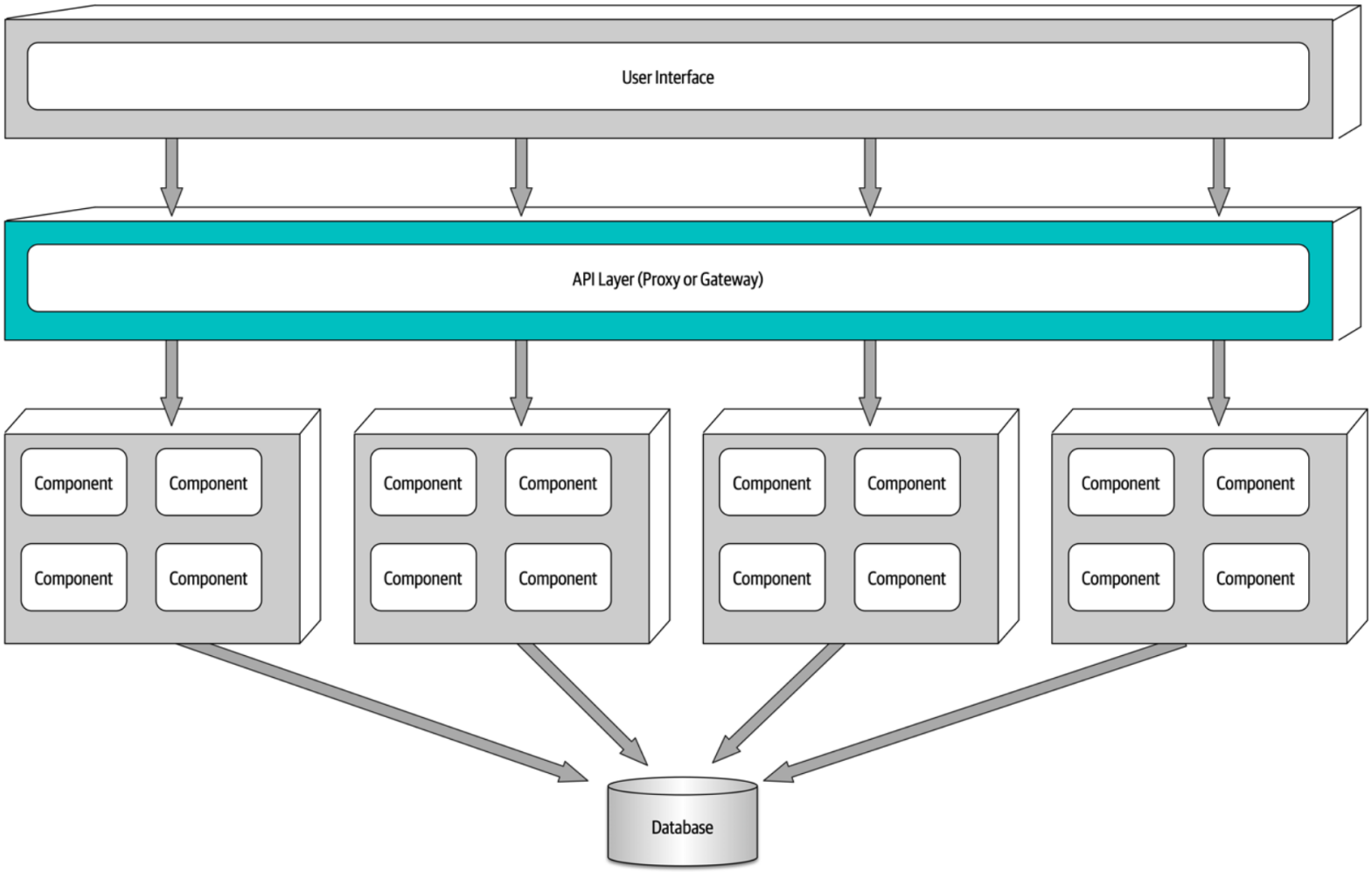


Figure 13-4. Adding an API layer between the user interface and domain services

Event-driven architecture

- Request based vs event based

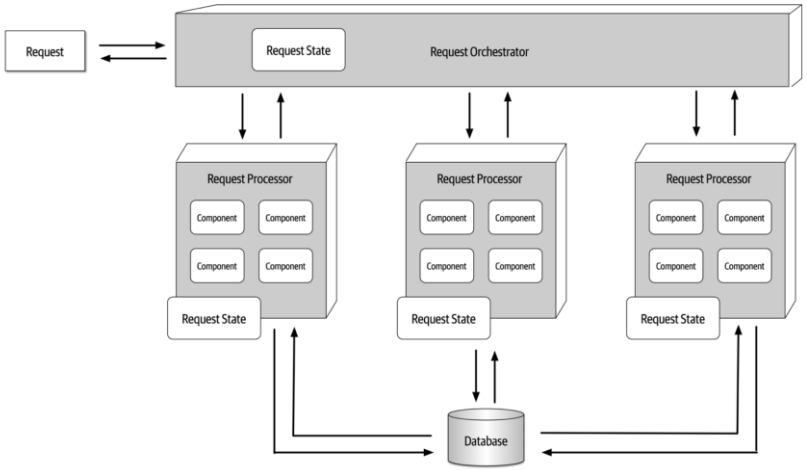


Figure 14-1. Request-based model

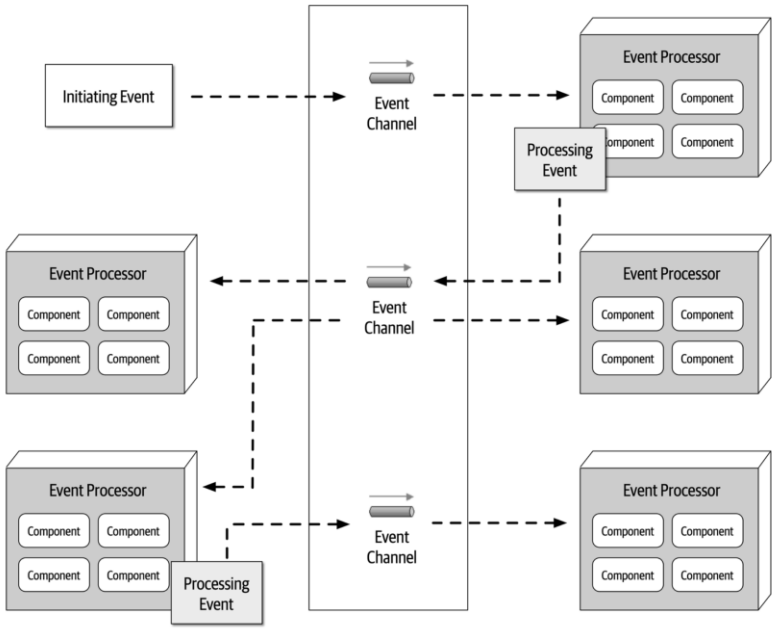


Figure 14-2. Broker topology

Event-driven architecture

Table 14-3. Trade-offs of the event-driven model

Advantages over request-based	Trade-offs
Better response to dynamic user content	Only supports eventual consistency
Better scalability and elasticity	Less control over processing flow
Better agility and change management	Less certainty over outcome of event flow
Better adaptability and extensibility	Difficult to test and debug
Better responsiveness and performance	
Better real-time decision making	
Better reaction to situational awareness	

Asynchronous

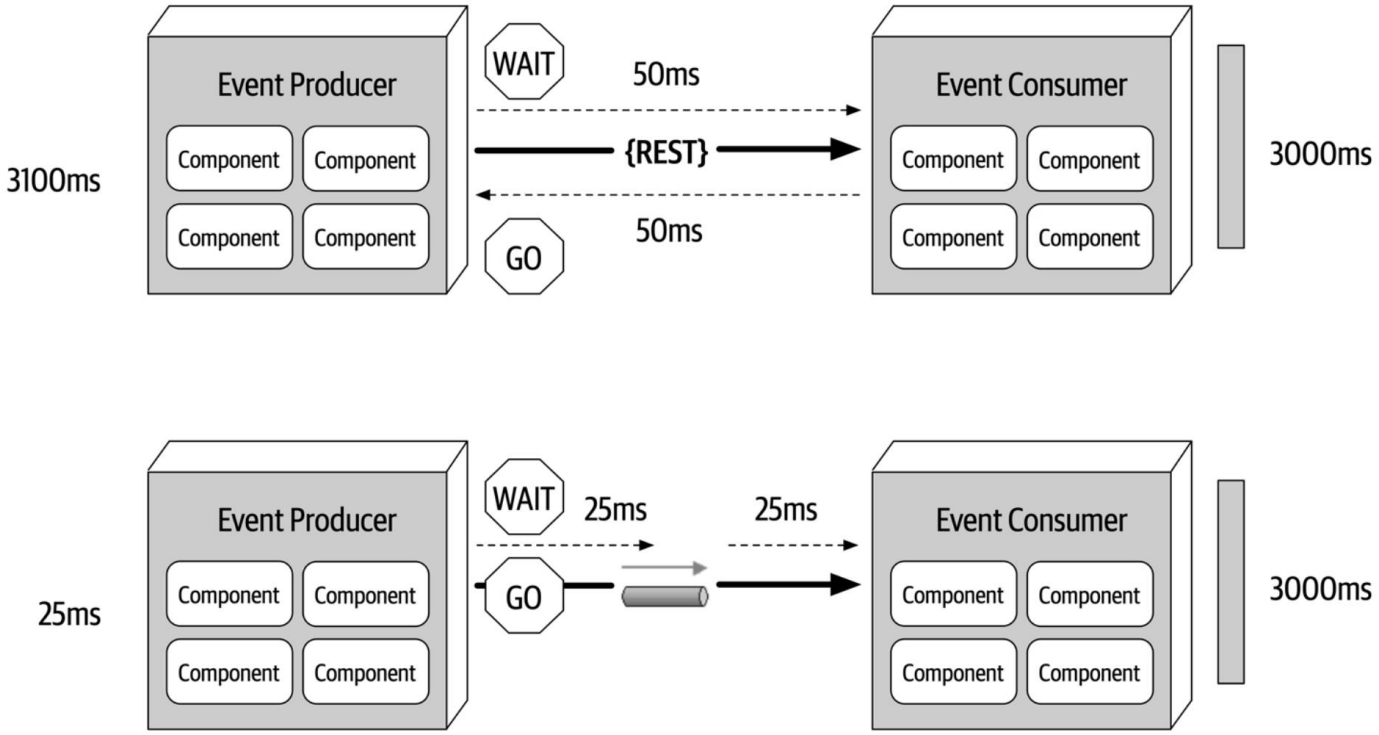


Figure 14-13. Synchronous versus asynchronous communication

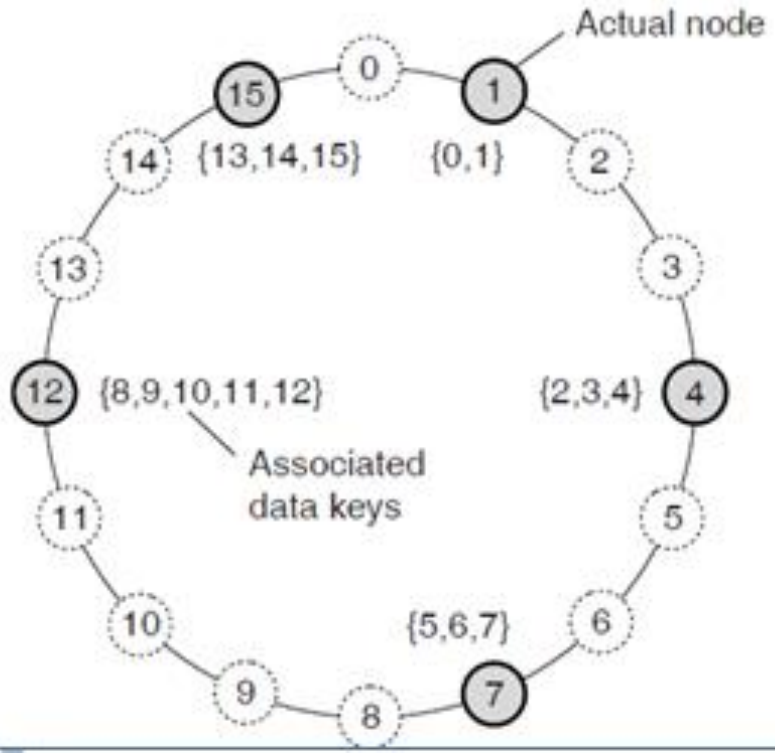
Decentralized Architecture

- Peer-to-peer systems banyak digunakan dalam berbagai aplikasi
  - Structured P2P: nodes terorganisasi berdasarkan struktur data terdistribusi tertentu
  - Unstructured P2P: nodes memilih neighbors secara random
  - Hybrid P2P: nodes memiliki fungsi khusus sesuai aturan tertentu

  - Jaringan Peer-to-peer umumnya memiliki overlay networks: ketetanggaan antar node didefinisikan berdasarkan aplikasi, bukan berdasarkan struktur fisik node

Structured P2P Systems

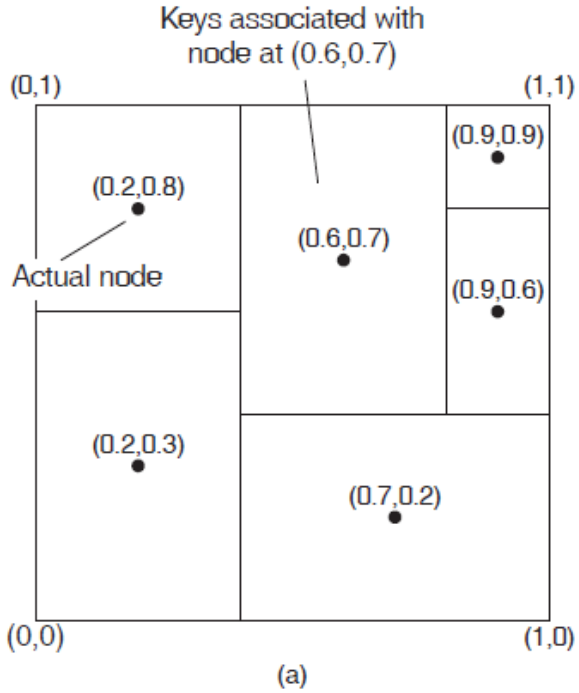
- Node diorganisasi berdasarkan model tertentu, misalnya ring, dan setiap node bertanggung jawab berdasarkan ID nya. Sistem menyediakan mekanisme yang memungkinkan mencari sebuah key berada pada node mana



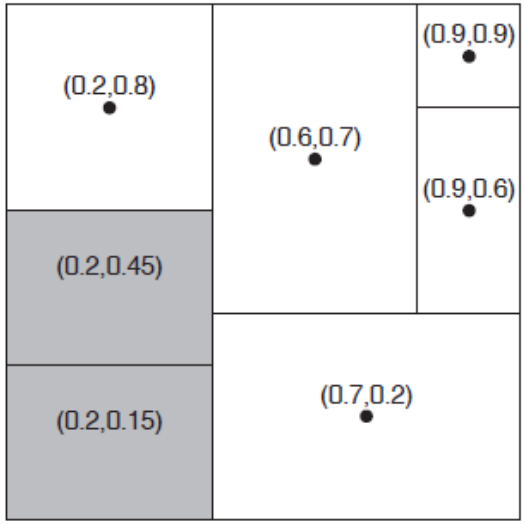
Structured P2P Systems

Other example

Organize nodes in a *d*-dimensional space and let every node take the responsibility for data in a specific region. When a node joins ⇒ split a region.



(a)



(b)

Unstructured P2P Systems

Observation

Many unstructured P2P systems attempt to maintain a random graph.

Basic principle

Each node is required to contact a randomly selected other node:

- Let each peer maintain a partial view of the network, consisting of c other nodes
- Each node P periodically selects a node Q from its partial view
- P and Q exchange information **and** exchange members from their respective partial views

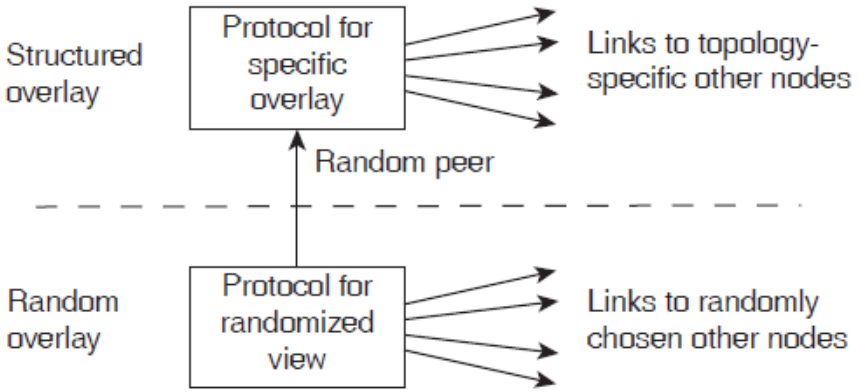
Note

It turns out that, depending on the exchange, randomness, but also robustness of the network can be maintained.

Topology Management of Overlay Networks

Basic idea

Distinguish two layers: (1) maintain random partial views in lowest layer; (2) be selective on who you keep in higher-layer partial view.



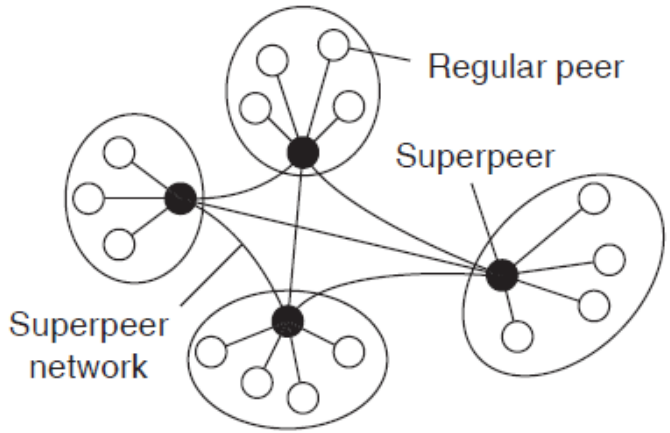
Note

Lower layer feeds upper layer with random nodes; upper layer is selective when it comes to keeping references.

Superpeers

Observation

Sometimes it helps to select a few nodes to do specific work: superpeer.



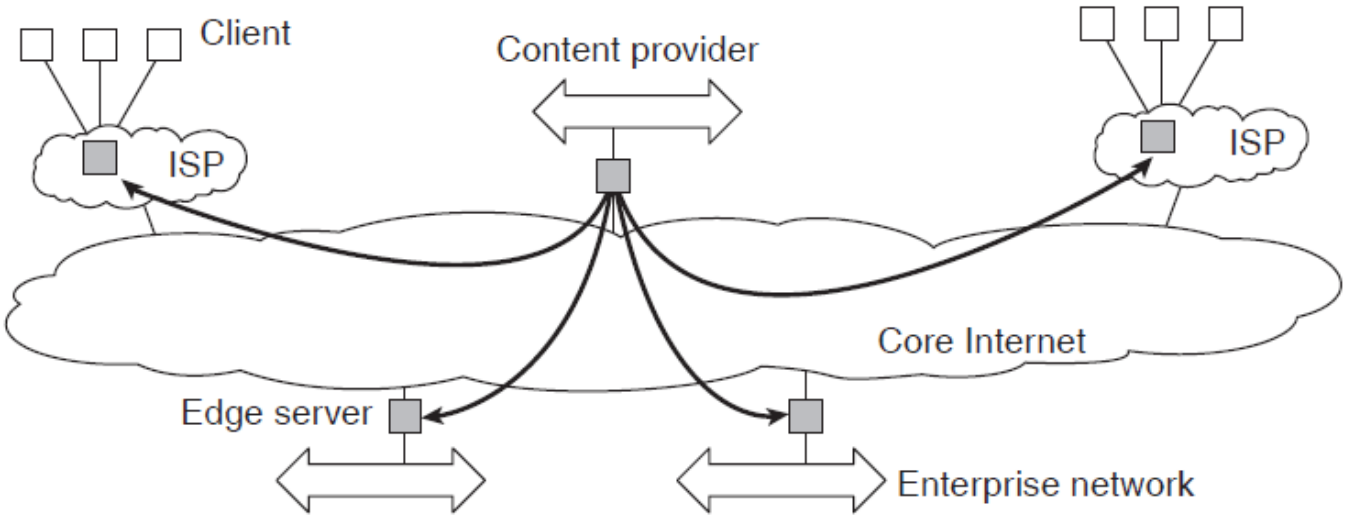
Examples

- Peers maintaining an index (for search)
- Peers monitoring the state of the network
- Peers being able to setup connections

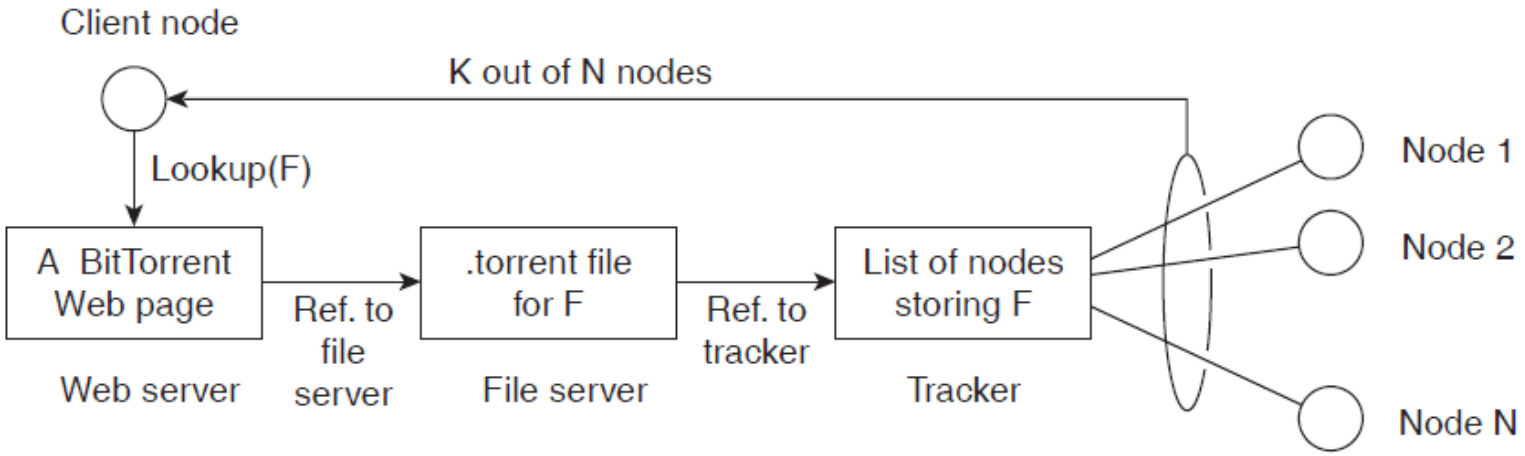
Hybrid Architectures: Client-servers + P2P

Example

Edge-server architectures, which are often used for Content Delivery Networks



Hybrid Architecture: CS with P2P - BitTorrent



Basic idea

Once a node has identified where to download a file from, it joins a swarm of downloaders who in parallel get file chunks from the source, but also distribute these chunks amongst each other.

Sumber

- Van Steen, Maarten, and Andrew S. Tanenbaum. *Distributed systems*. Leiden, The Netherlands: Maarten van Steen, 2017.
- Richards, Mark, and Neal Ford. *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, 2020.
- Vitillo, Roberto. *Understanding Distributed Systems.* 2021. https://understandingdistributed.systems/