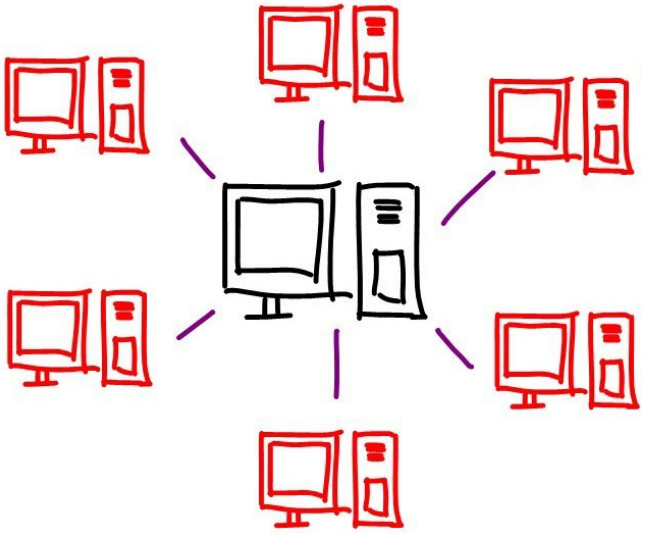
Sistem Terdistribusi (Distributed System)

Arsitektur sistem terdisribusi

Henry Saptono, M.Kom



Course/Slides Credits

- **Catatan**: *semua slide presentasi didasarkan pada buku yang dikembangkan oleh Andrew S. Tanenbaum dan Maarten van Steen. Judul buku mereka "Distributed Systems: Principles and Paradigms" (edisi 1 & 2). Dan juga berdasarkan slide presentasi yang dibuat oleh Maarten van Steen (VU Amsterdam, Dept. Computer Science)*

Pengantar

- Sistem terdistribusi seringkali merupakan bagian perangkat lunak yang **kompleks** yang menurut definisi, komponennya **tersebar** di beberapa mesin.
- Untuk mampu menguasai **kerumitannya**, sangat penting untuk memastikan bahwa sistem ini dikelola/ diatur dengan baik
- Ada berbagai cara tentang cara memandang atau melihat bagaimana organisasi dari sistem terdistribusi:
    - Organisasi logis → kumpulan komponen perangkat lunaknya,
    - Realisasi aktual fisiknya

Pengantar

- **Organisasi** sistem terdistribusi sebagian besar tentang komponen **perangkat lunak** yang **membentuk sistem**.
- Arsitektur perangkat lunak ini memberi tahu kita bagaimana berbagai komponen perangkat lunak **diatur** dan bagaimana mereka harus **berinteraksi**.

Aristektur

- Tujuan penting dari sistem terdistribusi adalah untuk memisahkan aplikasi dari platform yang mendasarinya dengan menyediakan lapisan **middleware.**
- Mengadopsi lapisan seperti itu adalah keputusan arsitektur yang penting, dan tujuan utamanya adalah untuk memberikan **transparansi distribusi**.

Aristektur

- Architectural Styles
- System Architecture

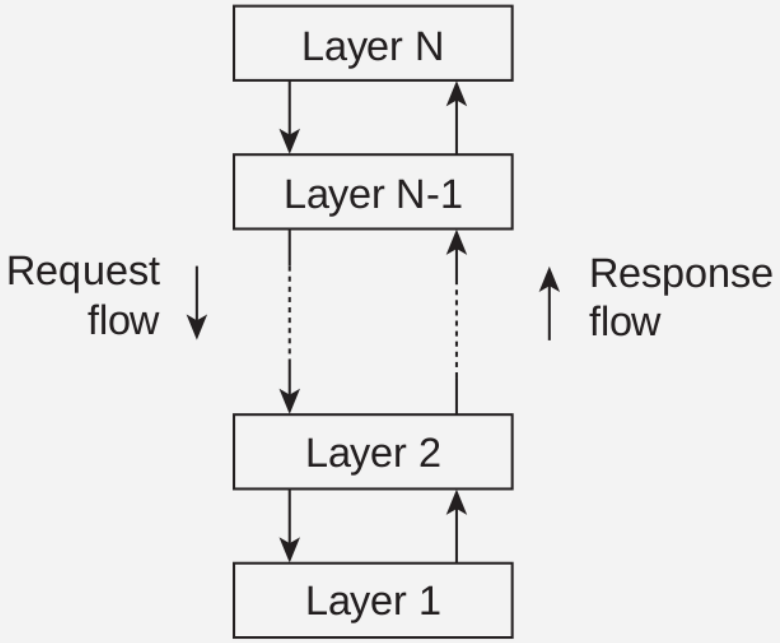
Architectural Styles

- Gaya (styles) dirumuskan dalam hal komponen, **cara komponen terhubung** satu sama lain, **pertukaran data** antar komponen, dan akhirnya bagaimana elemen-elemen ini secara bersama-sama **dikonfigurasikan** ke dalam suatu sistem
- Komponen adalah unit modular dengan antarmuka yang diperlukan dan disediakan dengan baik yang dapat diganti dalam lingkungannya [OMG, 2004b].
- Mempertimbangkan organisasi logis dari sistem terdistribusi ke dalam komponen perangkat lunak, juga disebut sebagai arsitektur perangkat lunaknya (**Software Architectures**)

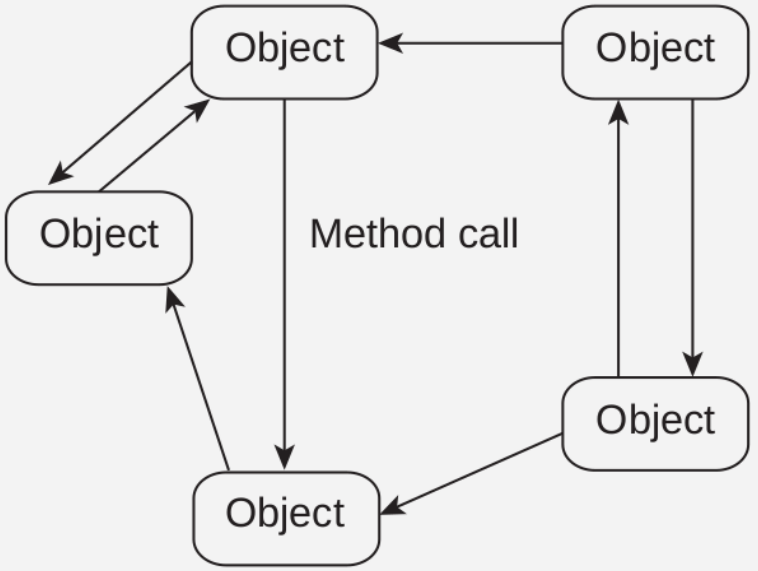
Gaya arsitektur (Architectural styles)

- Ide dasar:
  - Mengatur komponen yang berbeda secara logis, dan mendistribusikan komponen tersebut melalui berbagai mesin.
- Gaya arsitektur:
  - Layered architectures
  - Object-based architectures
  - Resource-centered architectures
  - Event-based architectures

Gaya arsitektur (Architectural styles)



(a)



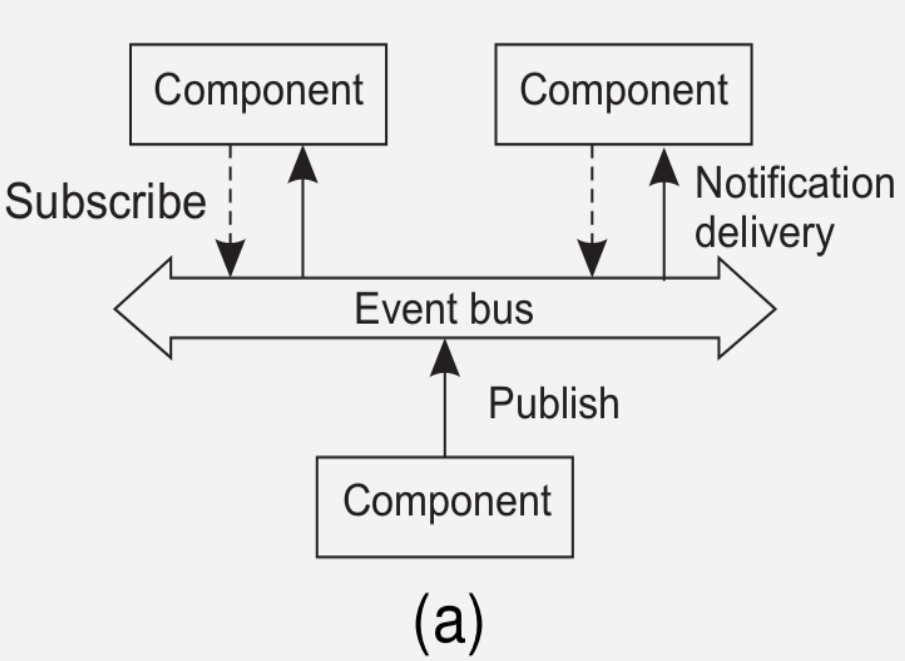
(b)

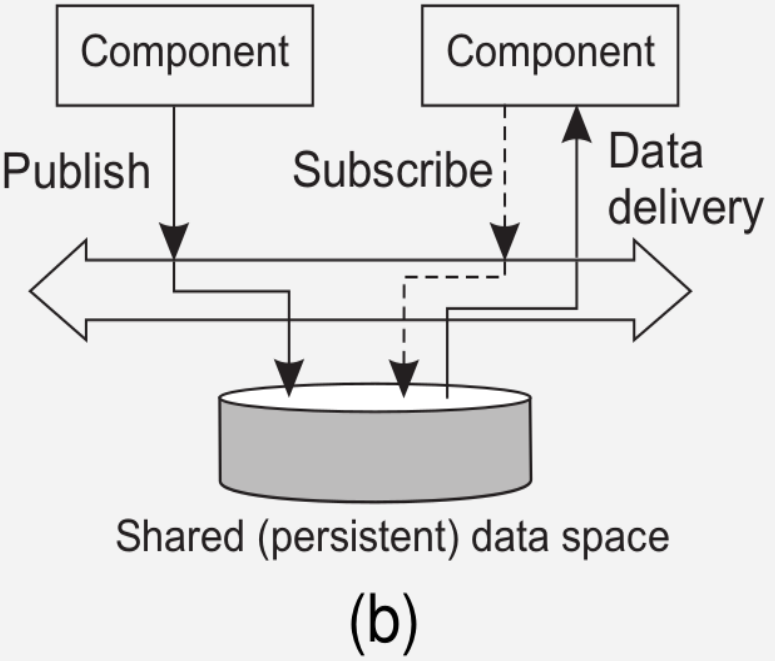
(a) Layered style is used for client-server system
(b) Object-based style for distributed object systems.

Gaya arsitektur (Architectural styles)

- Proses **decoupling ruang** ("anonymous") dan juga **waktu** ("asynchronous") telah mengarah ke gaya/style alternatif.

Gaya arsitektur (Architectural styles)



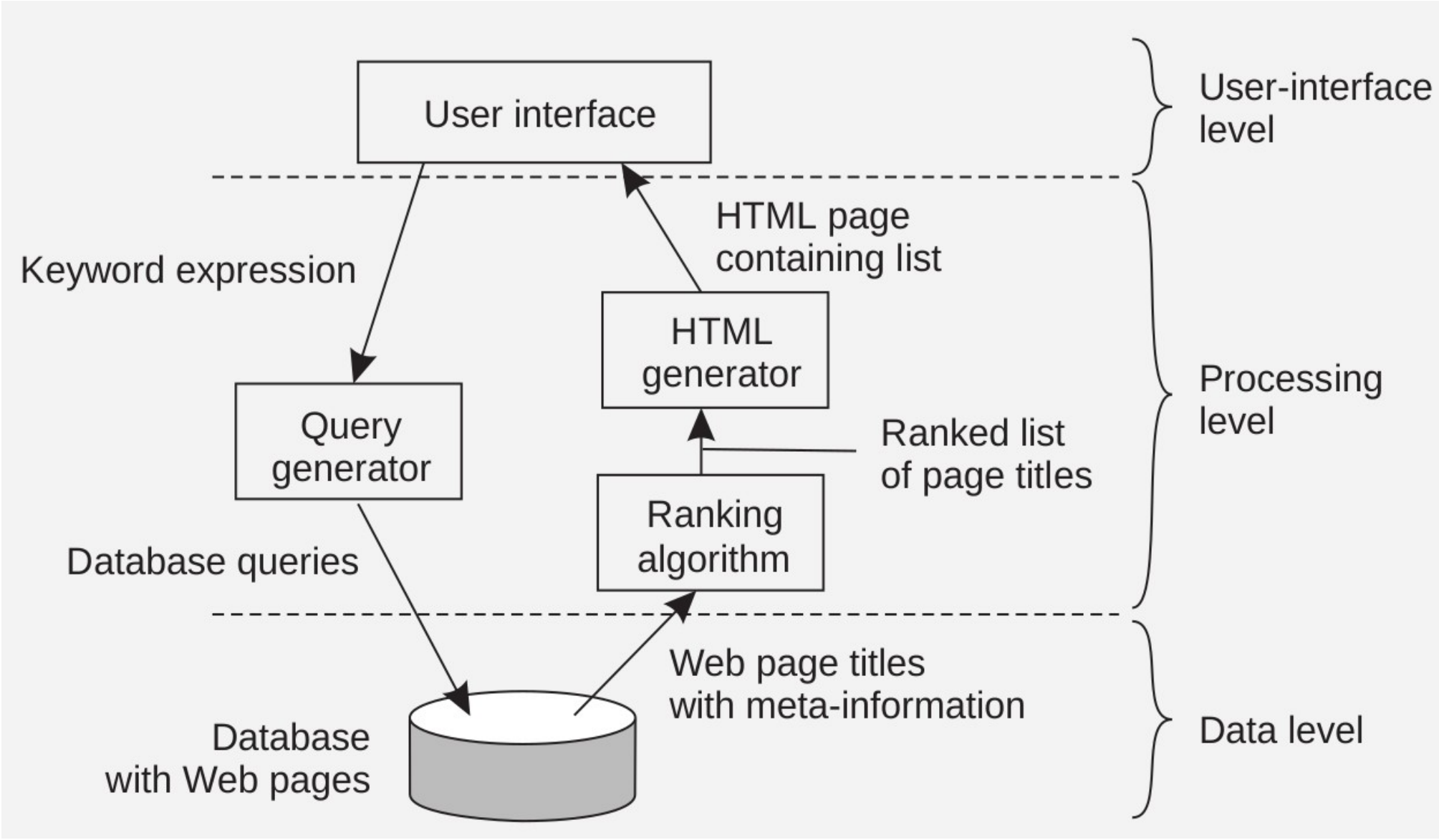


(a) Publish/subscribe [decoupled in space]
(b) Shared dataspace [decoupled in space and time]

Application Layering

- Traditional three-layered view
  - Lapisan antarmuka pengguna berisi unit untuk antarmuka pengguna aplikasi
  - Lapisan pemrosesan berisi fungsi aplikasi
  - Lapisan data berisi data yang ingin dimanipulasi klien melalui komponen aplikasi
- Lapisan ini ditemukan di banyak sistem informasi terdistribusi, menggunakan teknologi database tradisional dan aplikasi yang menyertainya.

Application Layering



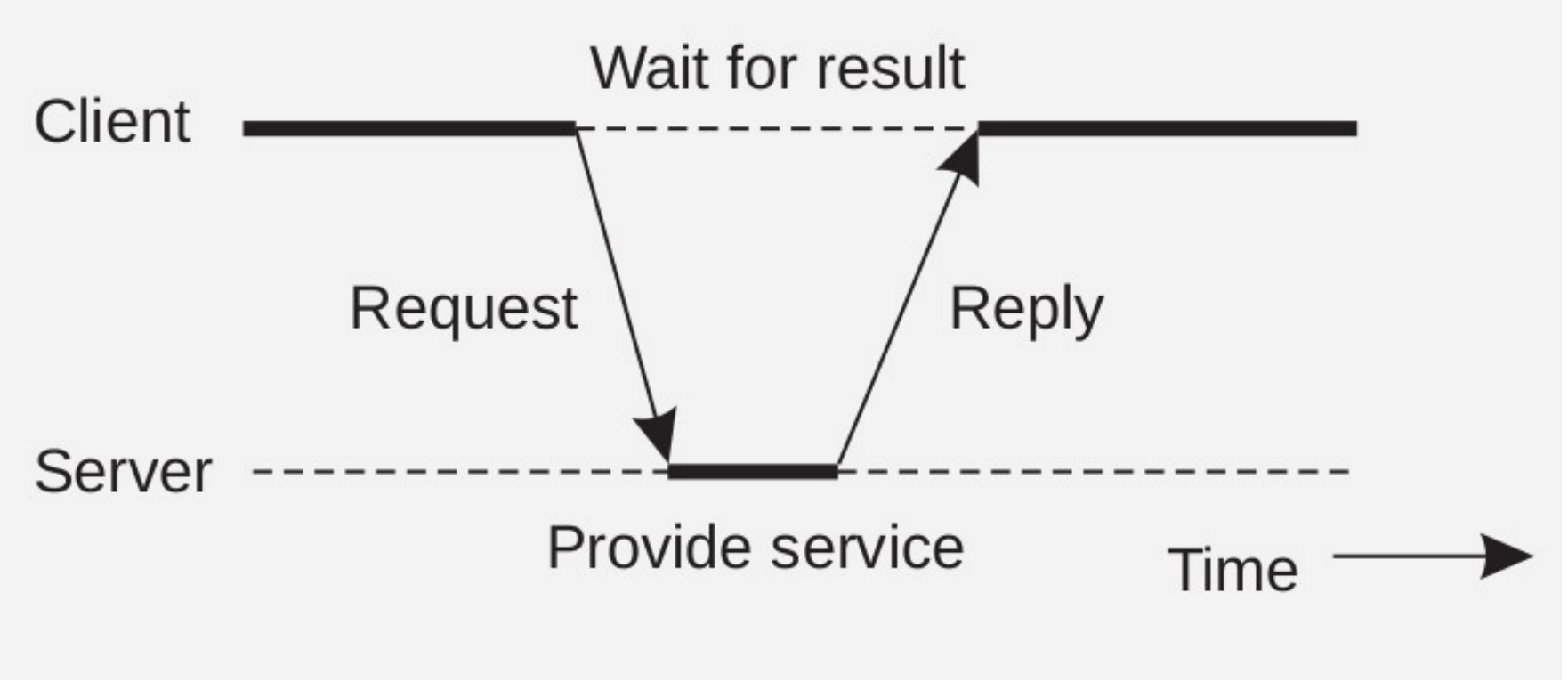
System Architectures

- Realisasi aktual dari sistem terdistribusi mengharuskan kita membuat instance dan **menempatkan komponen perangkat lunak pada mesin nyata**.
- Ada banyak pilihan berbeda yang dapat dilakukan untuk melakukannya. Instansiasi akhir **arsitektur perangkat lunak** juga disebut sebagai **arsitektur sistem.**

System Architecture - Arsitektur Terpusat

- **Basic Client–Server Model**
  - Karakteristik:
    - Ada proses yang menawarkan layanan (server)
    - Ada proses yang menggunakan layanan (klien)
    - Klien dan server dapat berada di mesin yang berbeda
    - Klien mengikuti model permintaan / balasan yang menggunakan layanan

Basic client server model



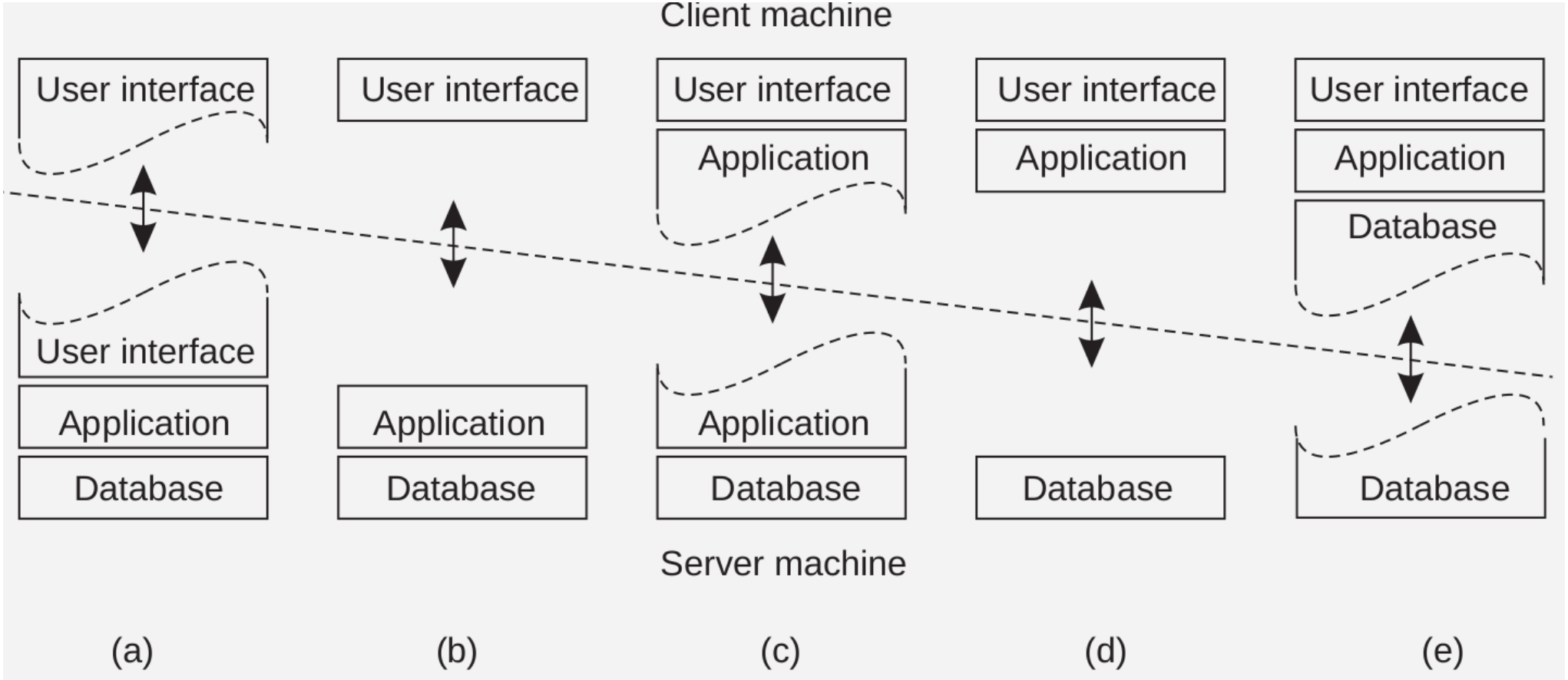
General interaction between a client and a server.

Multi-Tiered Architectures

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

Multi-Tiered Architectures

- Traditional two-tiered configurations:



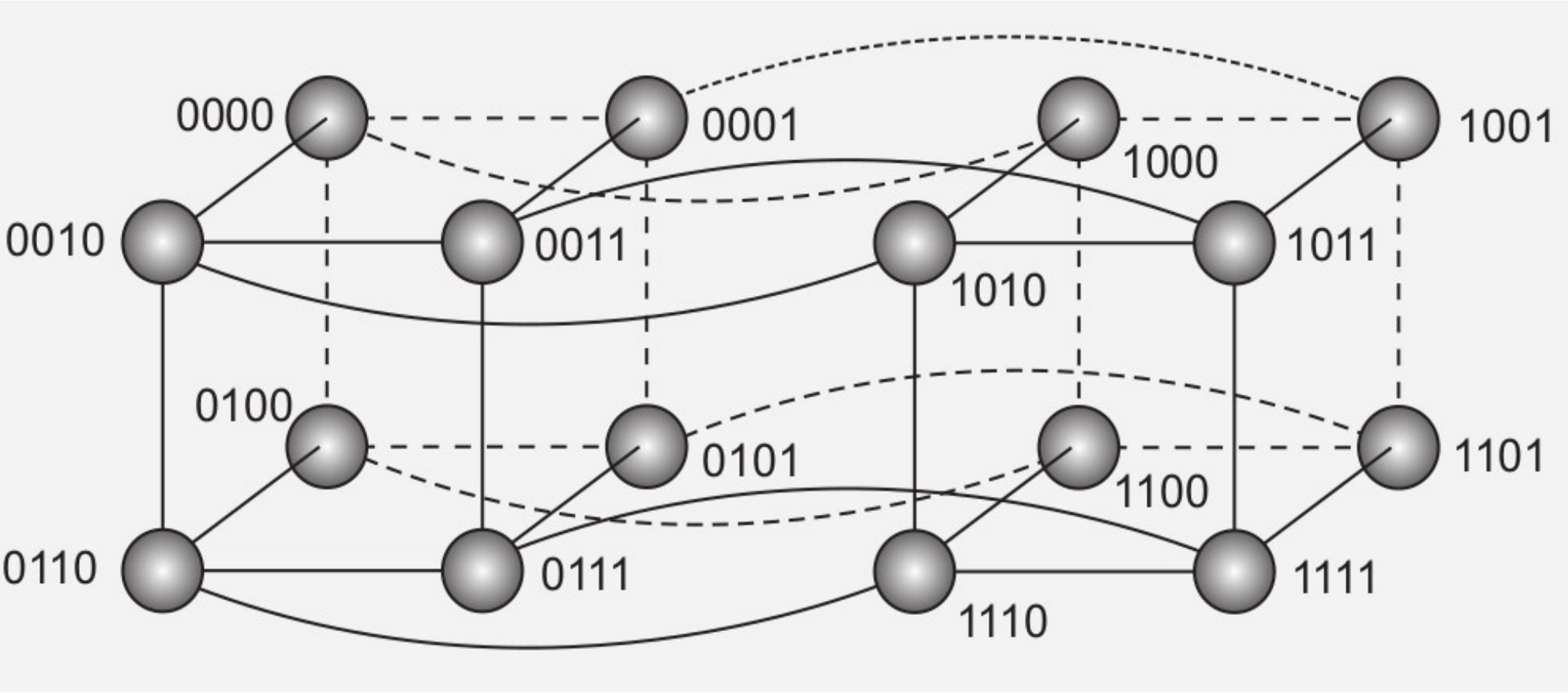
Arsitektur Terdesentralisasi

- Peer-to-peer (P2P).
  - P2P terstruktur: node disusun mengikuti struktur data terdistribusi tertentu
  - P2P tidak terstruktur: node memiliki tetangga yang dipilih secara acak
  - Hybrid P2P: beberapa node ditunjuk fungsi khusus dengan cara yang terorganisir dengan baik

Sistem P2P Terstruktur

- Ide dasar:
  - Mengatur node-node dalam jaringan hamparan terstruktur seperti cincin logis, atau hypercube, dan membuat node tertentu bertanggung jawab atas layanan yang hanya berdasarkan ID mereka.
- Catatan:
  - Sistem menyediakan operasi LOOKUP(key) yang secara efisien akan merutekan permintaan pencarian ke node terkait.

Sistem P2P Terstruktur



Sistem P2P Tidak Terstruktur

- Banyak sistem P2P tidak terstruktur diatur sebagai hamparan acak: dua node dihubungkan dengan probabilitas **p**
- Tidak lagi dapat mencari informasi secara deterministik, tetapi harus menggunakan pencarian:
  - **Flooding**: node **u** mengirimkan permintaan pencarian ke semua tetangganya. Tetangga merespons, atau meneruskan (membanjiri) permintaan. Ada banyak variasi:

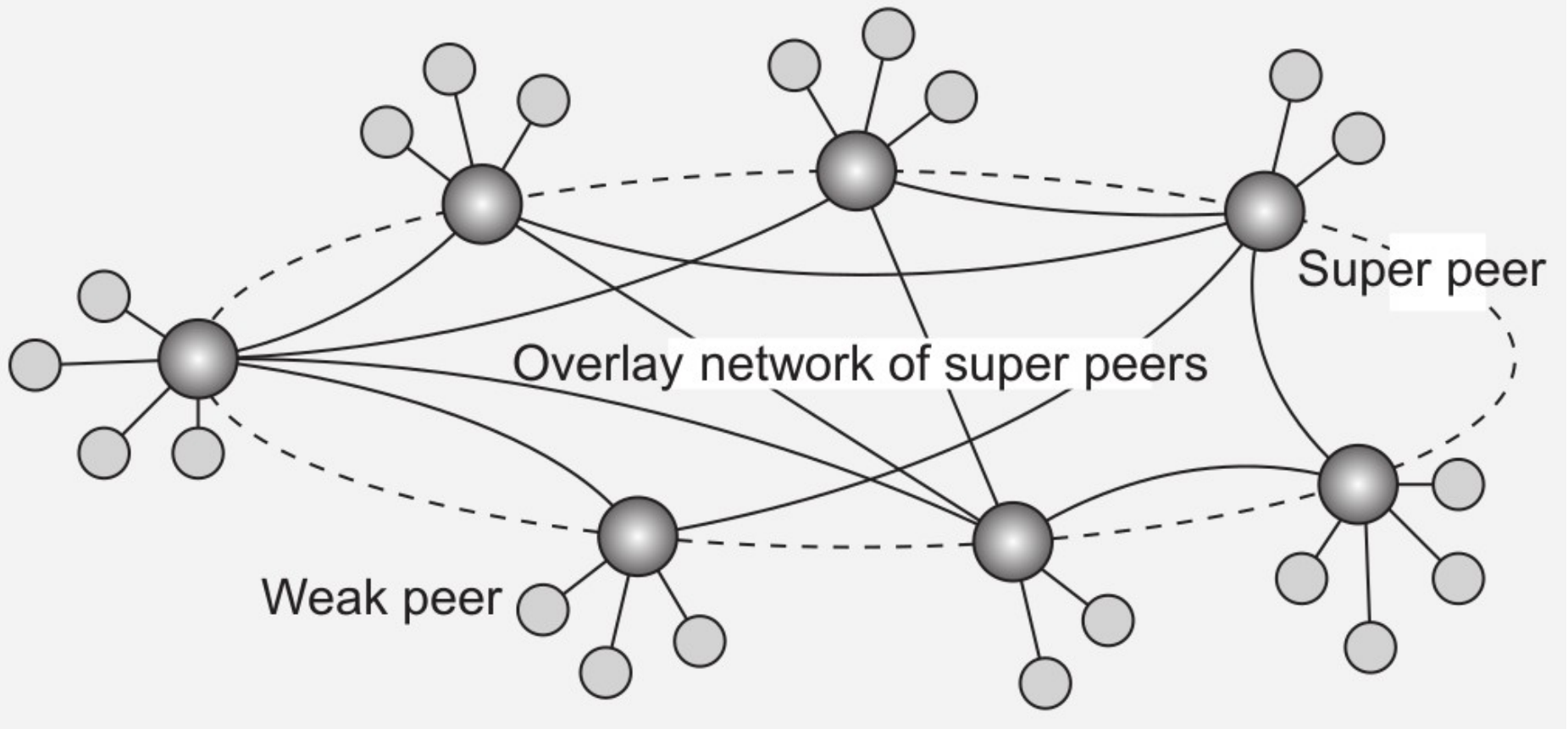
Sistem P2P Tidak Terstruktur

  - Limited Flooding (jumlah penerusan maksimal)
  - Probabilistic Flooding (hanya ‘banjir’ dengan probabilitas tertentu).
- Jalan acak (**Random Walk**): Pilih tetangga **v** secara acak. Jika **v** memiliki jawabannya, ia menjawab, jika tidak, **v** akan memilih salah satu tetangganya secara acak. Variasi: **parallel random walk**. Bekerja dengan baik dengan data yang direplikasi.

Superpeers

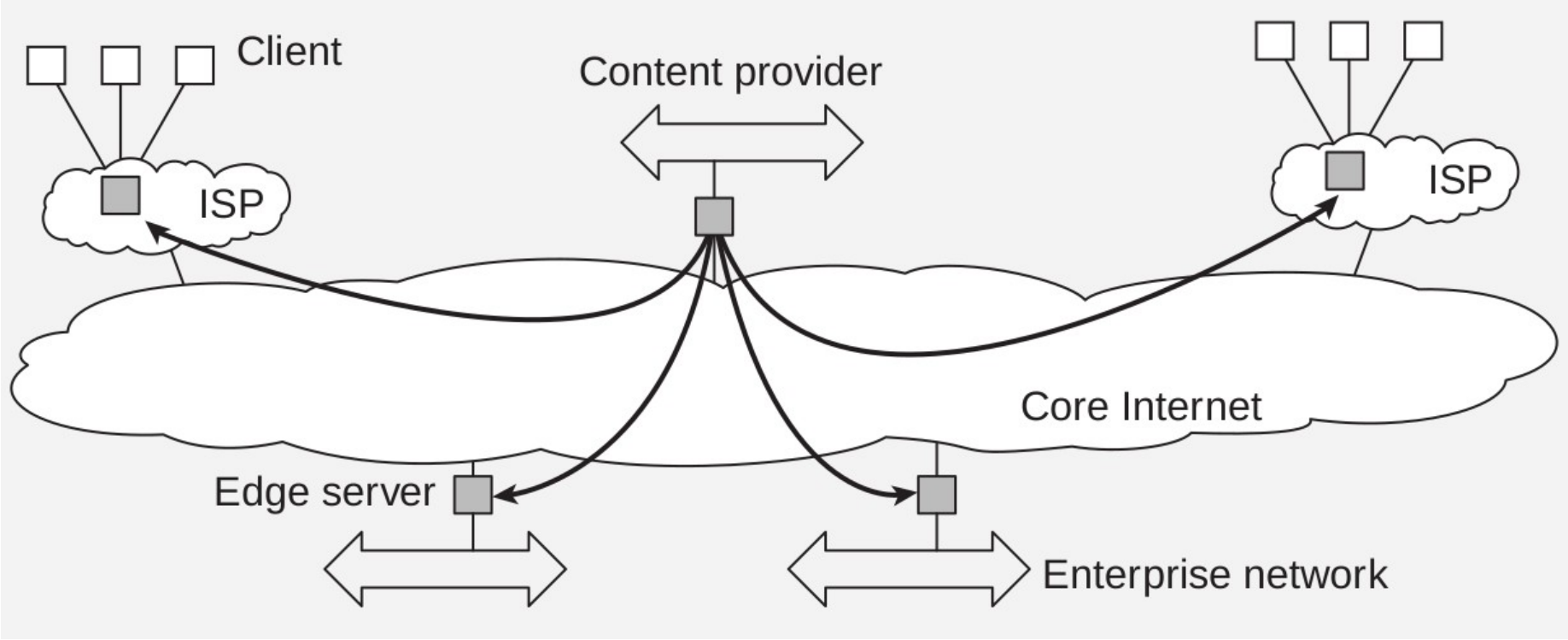
- Terkadang ini membantu untuk memilih beberapa node untuk melakukan pekerjaan tertentu: superpeer.
- Contoh:
  - Peer mempertahankan indeks (untuk pencarian)
  - Peer memonitor keadaan jaringan
  - Peer bisa mengatur koneksi

Superpeers

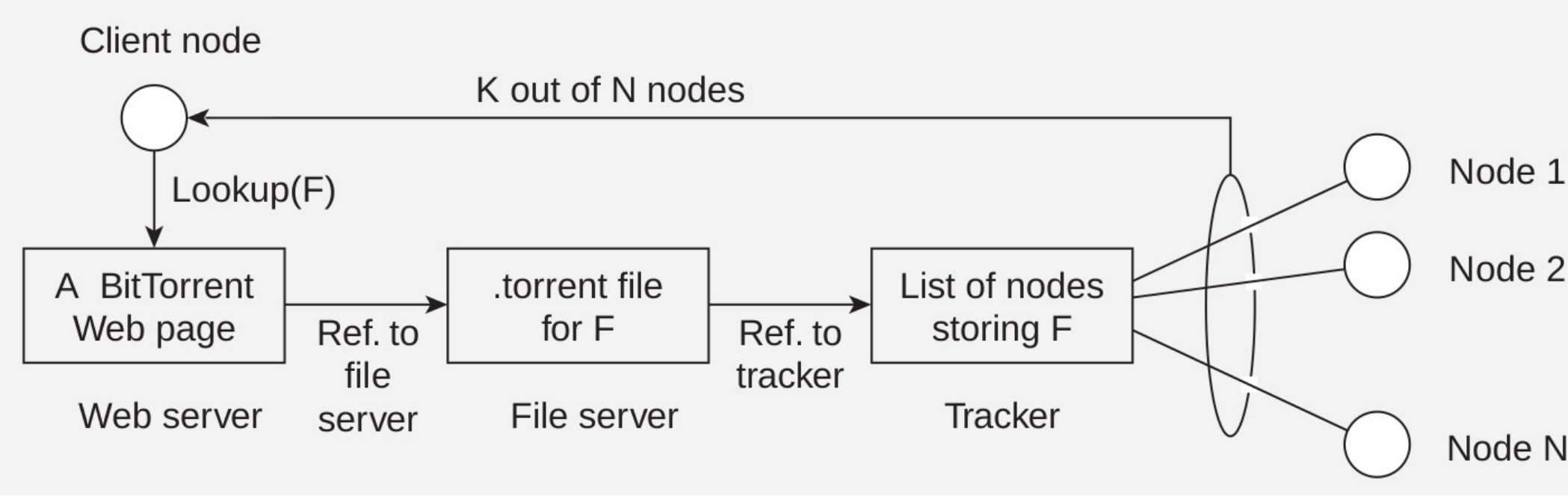


Hybrid Architectures: Client-server combined with P2P

- Contoh:
  - **Edge-server architectures**, yang sering digunakan untuk Jaringan Pengiriman Konten (*Content Delivery Networks*)



Hybrid Architectures: C/S with P2P – BitTorrent



- Ide dasar:

  Setelah sebuah node mengidentifikasi tempat untuk mengunduh file, ia bergabung dengan sekelompok pengunduh yang secara paralel mendapatkan potongan file dari sumbernya, tetapi juga mendistribusikan potongan-potongan ini di antara satu sama lain.

Arsitektur versus Middleware

- Dalam banyak kasus, sistem / aplikasi terdistribusi dikembangkan sesuai dengan gaya arsitektur tertentu. Gaya yang dipilih mungkin tidak optimal dalam semua kasus ⇒ perlu (secara dinamis) menyesuaikan perilaku middleware.