Pertemuan 2
Arsitektur Sistem Terdistribusi

Pokok Bahasan

- Arsitektur style
- Arsitektur berbasis object
- Arsitektur berbasis peristiwa
- System arsitektur
- Cetralize arsitektur
- Decentralize arsitektur
- Hybrid arsitektur

Arsitektur Style

Organisasi sistem terdistribusi sebagian besar tentang komponen perangkat lunak yang membentuk sistem. Arsitektur perangkat lunak ini memberi tahu bagaimana berbagai komponen perangkat lunak diatur dan bagaimana mereka harus berinteraksi. Dalam hal ini pertama-tama yang perlu diperhatikan adalah beberapa pendekatan yang biasa diterapkan untuk mengatur (mendistribusikan) sistem komputer.

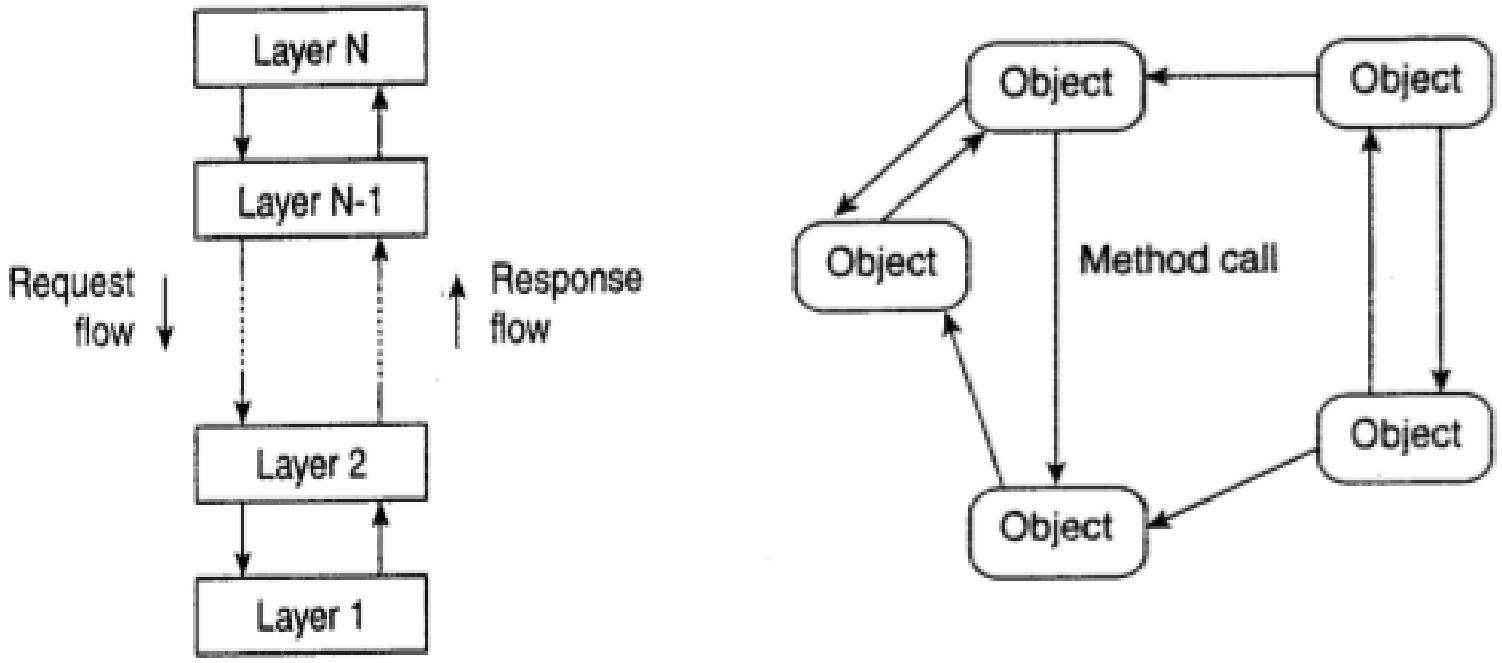
Realisasi aktual dari sistem terdistribusi diharuskam membuat instance dan menempatkan komponen perangkat lunak pada mesin nyata. Ada banyak pilihan berbeda yang dapat dilakukan untuk melakukannya. Arsitektur tradisional terpusat di mana satu server mengimplementasikan sebagian besar komponen perangkat lunak (dan dengan demikian fungsionalitas), sementara klien jarak jauh dapat mengakses server menggunakan sarana komunikasi sederhana. Selain itu, juga dipertimbangkan arsitektur terdesentralisasi di mana mesin lebih atau kurang memainkan peran yang sama, serta organisasi hibrida.

Menggunakan komponen dan konektor, maka dapat datang ke berbagai konfigurasi, yang pada dasarnya telah diklasifikasikan ke dalam gaya arsitektur. Beberapa gaya sekarang telah diidentifikasi, yang mana yang paling penting untuk sistem terdistribusi adalah:

1. Arsitektur berlapis
2. Arsitektur berbasis objek
3. Arsitektur berbasis data
4. Arsitektur berbasis peristiwa

Organisasi yang jauh lebih longgar diikuti dalam arsitektur berbasis objek, yang diilustrasikan pada Gambar 2-1. Pada dasarnya, setiap objek sesuai dengan apa yang telah didefinisikan sebagai komponen, dan komponen-komponen ini terhubung melalui mekanisme pemanggilan prosedur (jarak jauh). Tidak mengherankan, arsitektur perangkat lunak ini cocok dengan arsitektur sistem client-server yang sudah jelaskan di atas. Arsitektur berlapis dan berbasis objek masih membentuk gaya yang paling penting untuk sistem perangkat lunak besar (Bass et al., 2003).

Arsitektur berbasis Object

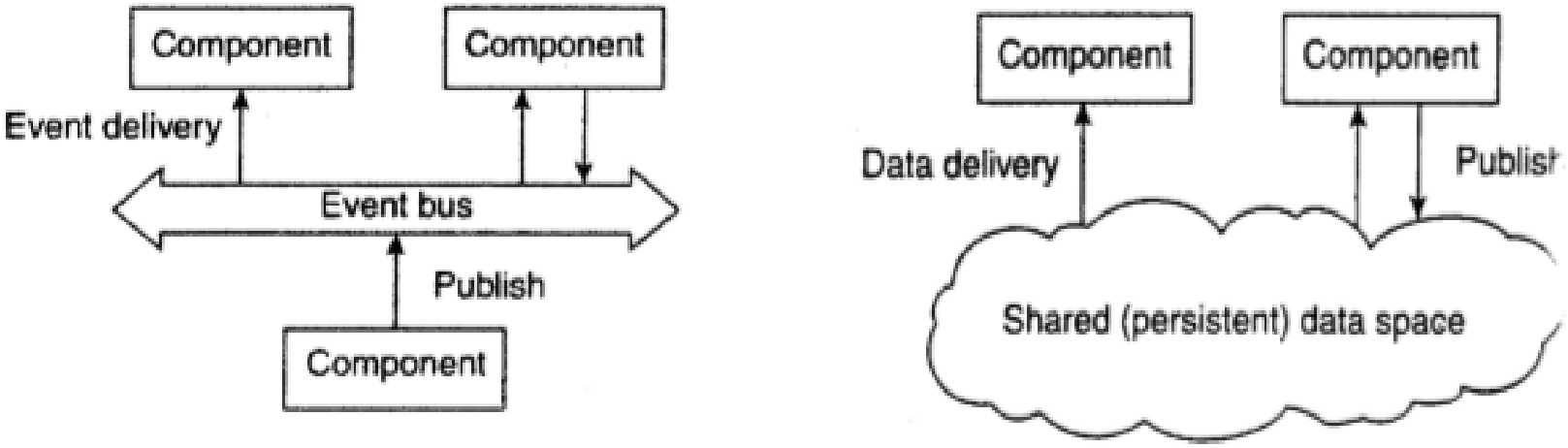


Gambar 2.1 Arsitektur berbasis Object

Arsitektur yang berpusat pada data berkembang di sekitar gagasan bahwa proses berkomunikasi melalui repositori umum (pasif atau aktif). Dapat dikatakan bahwa untuk sistem terdistribusi, arsitektur ini sama pentingnya dengan arsitektur berlapis dan berbasis objek. Sebagai contoh, banyak aplikasi jaringan telah dikembangkan yang mengandalkan sistem file terdistribusi bersama di mana hampir semua komunikasi terjadi melalui file. Demikian juga, sistem terdistribusi berbasis web, yang dibahas secara luas, sebagian besar data-sentris: proses berkomunikasi melalui penggunaan layanan data berbasis sharedWeb.

Dalam arsitektur berbasis peristiwa, proses pada dasarnya berkomunikasi melalui penyebaran peristiwa, yang secara opsional juga membawa data, seperti yang ditunjukkan pada Gambar. 2-2 (a). Untuk sistem terdistribusi, propagasi acara umumnya dikaitkan dengan apa yang dikenal sebagai sistem publish / subscribe (Eugster et al., 2003). Ide dasarnya adalah bahwa proses menerbitkan acara setelah middleware memastikan bahwa hanya proses yang berlangganan ke acara tersebut yang akan menerimanya. Keuntungan utama dari sistem berbasis peristiwa adalah bahwa proses digabungkan secara longgar. Pada prinsipnya, mereka tidak perlu saling merujuk secara eksplisit. Ini juga disebut sebagai dipisahkan dalam ruang, atau secara terpisah.

Arsitektur berbasis pristiwa



Gambar 2.2 Arsitektur berbasis Peristiwa

Arsitektur berbasis peristiwa dapat dikombinasikan dengan arsitektur berpusat data, menghasilkan apa yang juga dikenal sebagai ruang data bersama. Inti dari ruang data bersama adalah bahwa proses sekarang juga dipisahkan dalam waktu: mereka tidak perlu keduanya aktif ketika komunikasi terjadi. Selain itu, banyak ruang data bersama menggunakan antarmuka seperti SQL ke repositori bersama dalam arti bahwa data dapat diakses menggunakan deskripsi daripada referensi eksplisit, seperti halnya dengan file yang mengabdikan Chap. Pahami gaya arsitektur ini.

Apa yang membuat arsitektur perangkat lunak ini penting untuk sistem terdistribusi adalah bahwa mereka semua bertujuan untuk mencapai (pada tingkat yang wajar) transparansi distribusi. Namun, seperti yang telah sudah disampaikan, transparansi distribusi memerlukan pertukaran antara kinerja, toleransi kesalahan, kemudahan pemrograman, dan sebagainya. Karena tidak ada solusi tunggal yang akan memenuhi persyaratan untuk aplikasi terdistribusi yang memungkinkan, para peneliti telah meninggalkan gagasan bahwa sistem terdistribusi tunggal dapat digunakan untuk menutupi 90% dari semua kasus yang mungkin terjadi

SYSTEM ARCHITECTURES

Setelah membahas secara singkat beberapa gaya arsitektur umum, maka dapat dilihat bagaimana banyak sistem terdistribusi sebenarnya diatur dengan mempertimbangkan di mana komponen perangkat lunak ditempatkan. Memutuskan komponen perangkat lunak, interaksinya, dan penempatannya menyebabkan 10 contoh arsitektur perangkat lunak, juga disebut arsitektur sistem (Bass et al., 2003). Maka akan dibahas organisasi yang tersentralisasi dan terdesentralisasi, serta berbagai bentuk hibrida.

1. Centralized Architectures

Meskipun kurangnya konsensus tentang banyak masalah sistem terdistribusi, ada satu masalah yang disetujui oleh banyak peneliti dan praktisi: berpikir dalam hal klien yang meminta layanan dari server membantu memahami dan mengelola kompleksitas sistem terdistribusi dan itu adalah hal yang baik.

Dalam model client-server dasar, proses dalam sistem terdistribusi dibagi menjadi dua kelompok (mungkin tumpang tindih). Server adalah sebuah proses yang mengimplementasikan layanan tertentu, misalnya, layanan sistem file atau layanan basis data. Klien adalah proses yang meminta layanan dari server dengan mengirimkannya permintaan dan kemudian menunggu balasan server. Interaksi klien-server ini, juga dikenal sebagai perilaku balasan-pertanyaan ditunjukkan pada Gambar 2-3

Komunikasi antara klien dan server dapat diimplementasikan dengan menggunakan protokol tanpa koneksi sederhana ketika jaringan yang mendasarinya cukup dapat diandalkan seperti banyak jaringan area lokal. Dalam kasus ini, ketika klien meminta layanan, itu hanya paket pesan untuk server, mengidentifikasi layanan yang diinginkan, bersama dengan data input yang diperlukan. Pesan tersebut kemudian dikirim ke server. Yang terakhir, pada gilirannya, akan selalu menunggu permintaan masuk, selanjutnya memprosesnya, dan mengemas hasilnya dalam pesan yang kemudian dikirim kepada klien

Application Layering

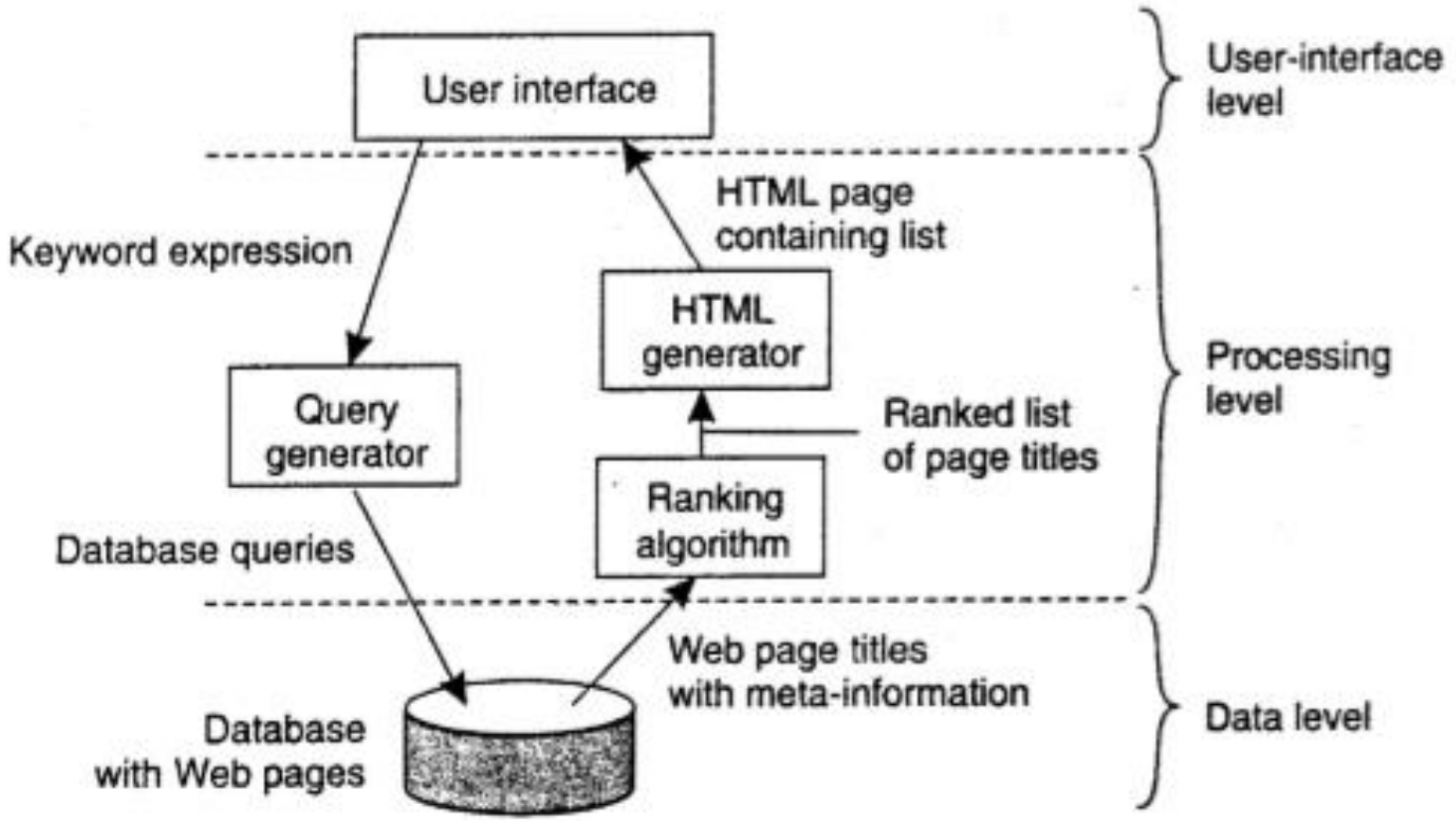
Model client-server telah mengalami banyak perdebatan dan kontroversi selama bertahun-tahun. Salah satu masalah utama adalah cara menggambar perbedaan yang jelas antara klien dan server. Tidak mengherankan, sering kali tidak ada perbedaan yang jelas. Misalnya, server untuk database terdistribusi dapat terus bertindak sebagai klien karena meneruskan permintaan ke server file yang berbeda yang bertanggung jawab untuk mengimplementasikan tabel database.

Dalam kasus seperti itu, server database itu sendiri pada dasarnya tidak lebih dari memproses permintaan. Namun, mengingat bahwa banyak aplikasi klien-server ditargetkan untuk mendukung akses pengguna ke database, banyak orang menganjurkan perbedaan antara tiga tingkat berikut, pada dasarnya mengikuti gaya arsitektur berlapis yang sudah dibahas sebelumnya:

1. Tingkat antarmuka pengguna
2. Tingkat pemrosesan
3. Tingkat data

Level antarmuka pengguna berisi semua yang diperlukan untuk berinteraksi langsung dengan pengguna, seperti manajemen tampilan. Level pemrosesan biasanya berisi aplikasi. Tingkat data mengelola data aktual yang sedang ditindaklanjuti. Klien biasanya menerapkan level antarmuka pengguna. Level ini terdiri dari program yang memungkinkan pengguna akhir untuk berinteraksi dengan aplikasi. Program antarmuka pengguna yang paling sederhana tidak lebih dari layar berbasis karakter. Antarmuka seperti itu biasanya digunakan di lingkungan mainframe. Dalam kasus-kasus di mana mainframe mengontrol semua interaksi, termasuk keyboard dan monitor, orang hampir tidak dapat berbicara tentang lingkungan client-server.

Pertimbangkan mesin pencari Internet. Mengabaikan semua spanduk animasi, gambar, dan pembungkus jendela mewah lainnya, antarmuka pengguna mesin pencari sangat sederhana: pengguna mengetikkan untaian kata kunci dan kemudian disajikan dengan daftar judul halaman Web. Bagian belakang dibentuk oleh basis data besar halaman Web yang telah dibuat sebelumnya dan diindeks. Inti dari mesin pencari adalah program yang mengubah string kata kunci pengguna menjadi satu atau lebih query database. Selanjutnya peringkat hasil menjadi daftar, dan mengubah daftar itu menjadi serangkaian halaman HTML. Di dalam model klien-server, bagian pengambilan informasi ini biasanya ditempatkan pada tingkat pemrosesan. Gambar 2-4 menunjukkan organisasi ini.



Gambar 2.4 Organisasi yang disederhanakan dari mesin pencari Internet menjadi tiga lapisan yang berbeda.

System Arsitektur

Perbedaan menjadi tiga level logis seperti yang dibahas sejauh ini, menunjukkan sejumlah kemungkinan untuk secara fisik mendistribusikan aplikasi client-server di beberapa mesin. Organisasi yang paling sederhana hanya memiliki dua jenis mesin:

1. Mesin klien yang hanya berisi program yang mengimplementasikan (bagian dari) level antarmuka pengguna
2. Mesin server yang berisi sisanya, yaitu program yang menerapkan level pemrosesan dan data.

Dalam organisasi ini semuanya ditangani oleh server sementara klien pada dasarnya tidak lebih dari dum terminal, mungkin dengan antarmuka yang cukup grafis.

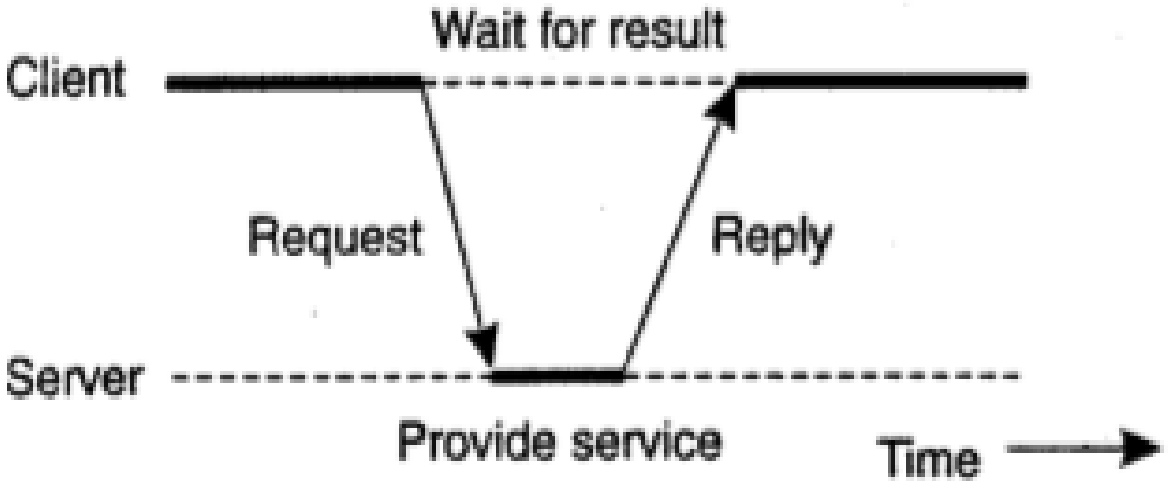
Dalam arsitektur ini, program yang membentuk bagian dari tingkat pemrosesan berada pada server terpisah, tetapi juga dapat didistribusikan sebagian di seluruh mesin klien dan server. Contoh khas di mana arsitektur tiga tingkat digunakan adalah

System Arsitektur

1 Arsitektur Terpusat

Meskipun kurangnya konsensus tentang banyak masalah sistem terdistribusi, ada satu masalah yang disetujui oleh banyak peneliti dan praktisi: berpikir dalam hal klien yang meminta layanan dari server membantu memahami dan mengelola kompleksitas sistem terdistribusi dan itu adalah hal yang baik.

Centralize Arsitektur



Gambar 2.3 Centralize Arsitektur

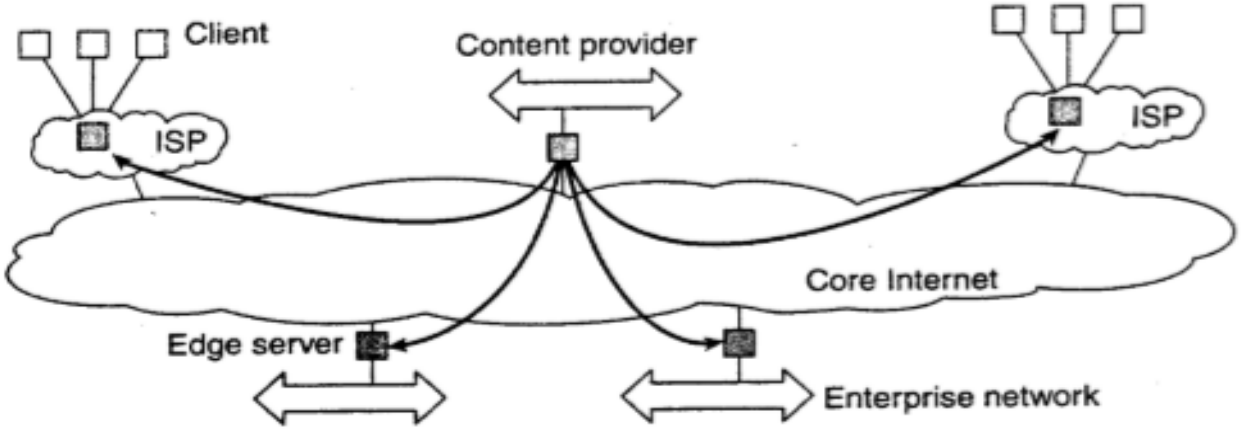
2. Decentralized Architectures

Arsitektur server klien multitier adalah konsekuensi langsung dari membagi aplikasi menjadi antarmuka pengguna, komponen pemrosesan, dan tingkat data. Tingkatan yang berbeda berhubungan langsung dengan pengaturan aplikasi yang logis. Di banyak lingkungan bisnis, pemrosesan terdistribusi setara dengan mengatur aplikasi klien-server sebagai arsitektur multitier. Penyebutan jenis distribusi ini sebagai distribusi vertikal. Ciri khas distribusi vertikal adalah dapat dicapai dengan menempatkan komponen yang berbeda secara logis pada mesin yang berbeda. Istilah ini terkait dengan konsep fragmentasi vertikal seperti yang digunakan dalam database relasional terdistribusi, di mana ini berarti bahwa tabel dibagi berdasarkan kolom, dan kemudian didistribusikan di beberapa mesin.

3. Hybrid Architectures

Sejauh ini, fokus pada arsitektur client-server dan sejumlah arsitektur peer-peer. Banyak sistem terdistribusi menggabungkan fitur arsitektur, seperti yang sudah ditemui pada jaringan superpeer. Pada bagian ini dapat dilihat beberapa kelas spesifik dari sistem terdistribusi di mana solusi client-server dikombinasikan dengan arsitektur terdesentralisasi

Kelas penting dari sistem terdistribusi yang diatur menurut arsitektur hibrid dibentuk oleh sistem edge-server. Sistem-sistem ini digunakan di Internet di mana server ditempatkan "di tepi" jaringan. Tepi ini dibentuk oleh batas antara jaringan perusahaan dan Internet yang sebenarnya, misalnya, sebagaimana disediakan oleh Penyedia Layanan Internet (ISP). Demikian juga, di mana pengguna akhir di rumah terhubung ke Internet melalui ISP mereka, ISP dapat dianggap berada di ujung Internet



Gambar 5. Melihat Internet sebagai terdiri dari kumpulan server tepi.