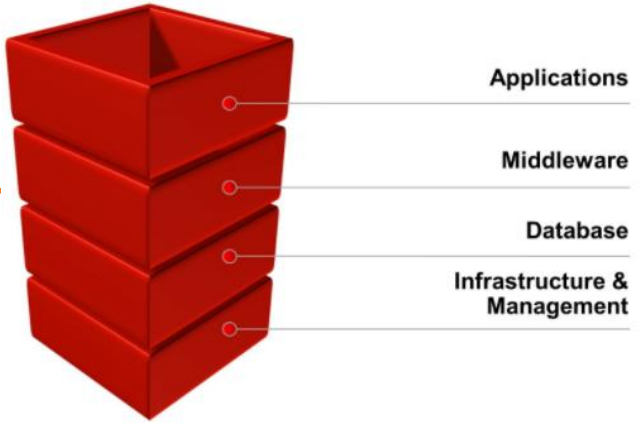
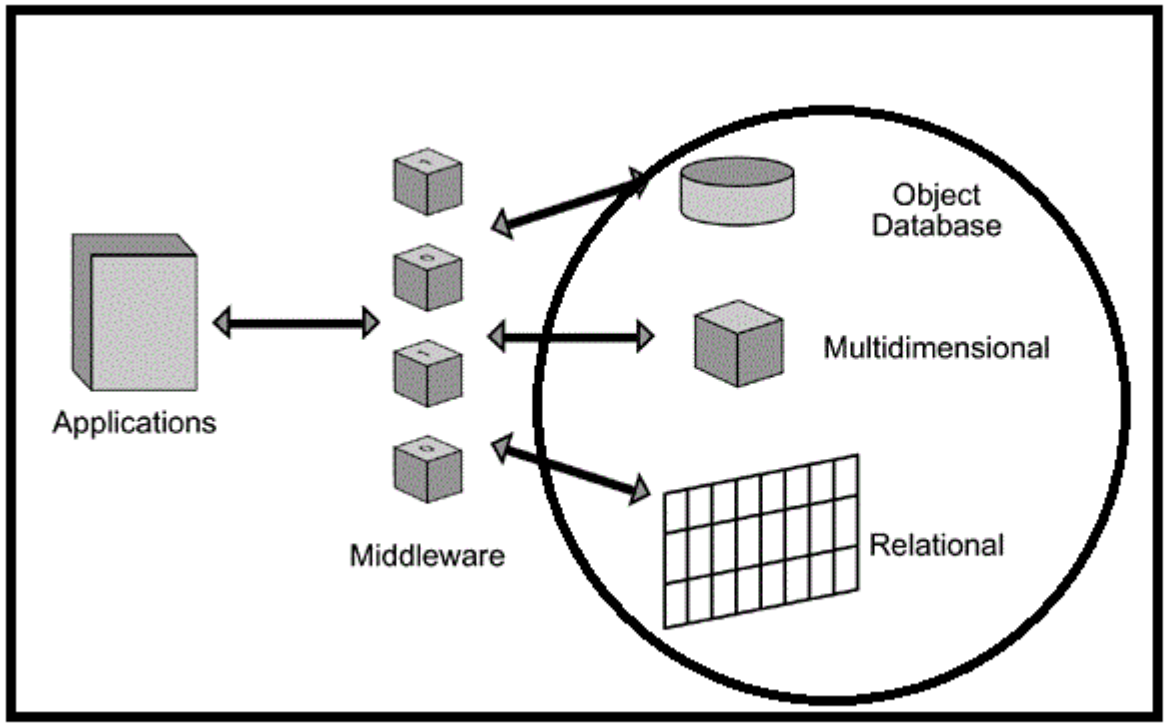
Integrasi Sistem
(Enterprise Application Integration)

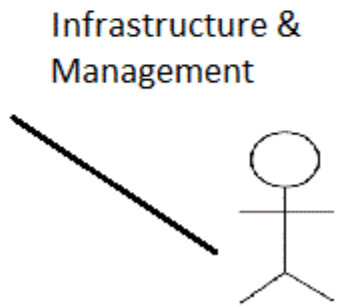
Sirojul Munir | rojulman@nurulfikri.ac.id

Level Integrasi

1. Level Database
2. Level Application
3. Level Middleware
4. Level Infrastructure & Management





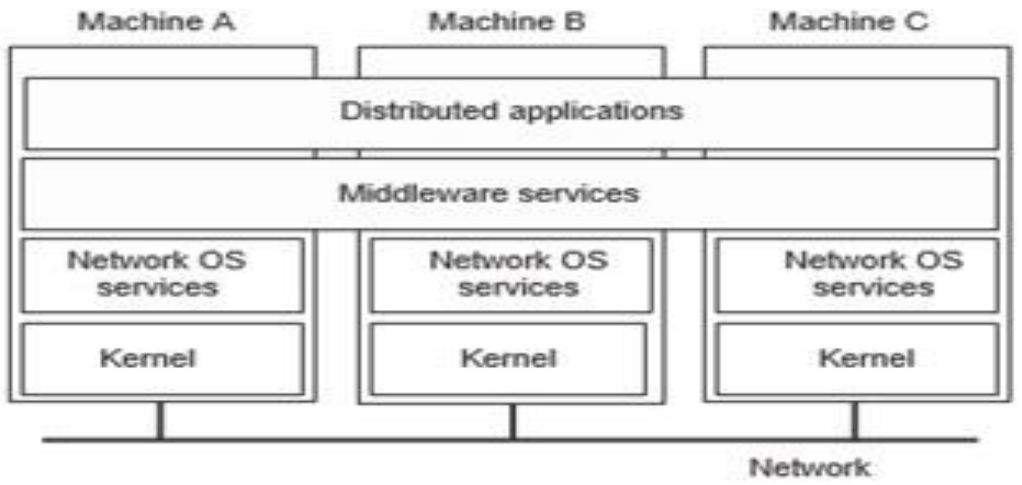


Integrasi Migrasi Level Middleware

Sirojul Munir | rojulman@nurulfikri.ac.id

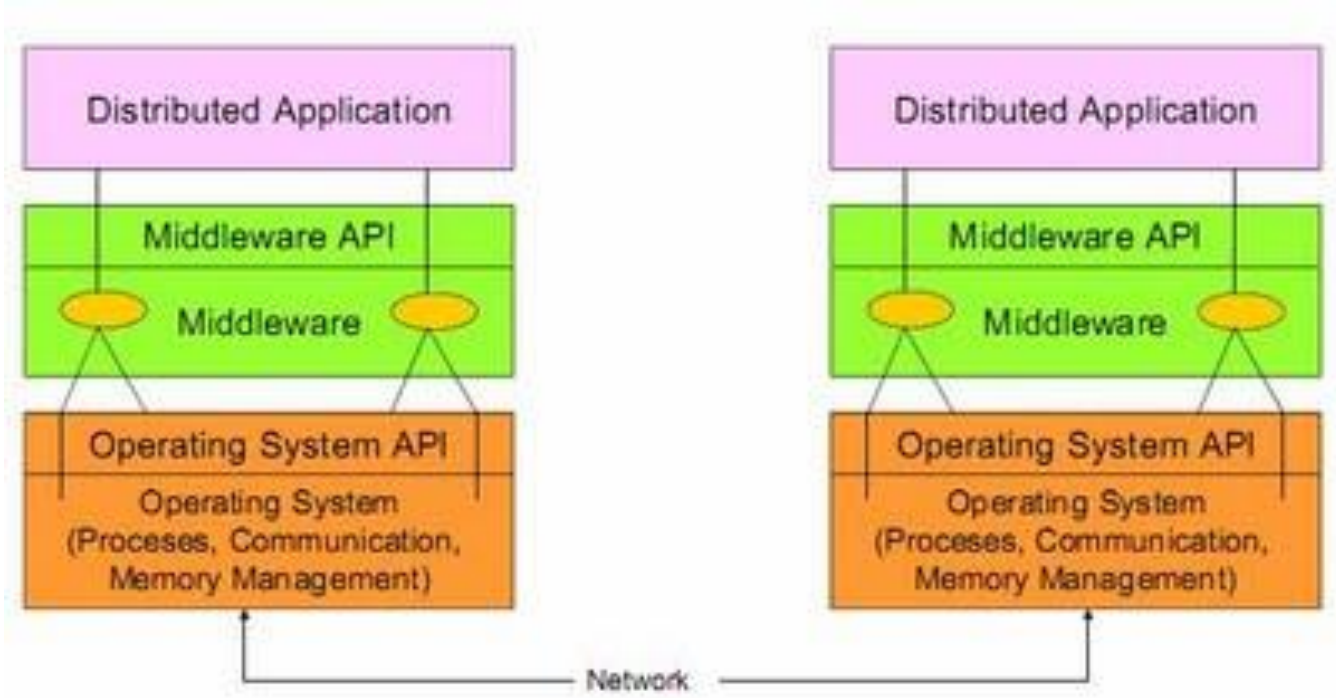
Definisi Middleware (1)

- Perangkat lunak (software) yang berfungsi sebagai penghubung (connector) dan memfasilitasi proses integrasi aplikasi-aplikasi yang berjalan pada lapisan atas (Application Layer, Translation Layer) dengan perangkat keras (hardware) yang berjalan di lapisan bawah, dimana penghubung keduanya adalah sistem operasi (putu agus-2016)



Definisi Middleware (2)

- Perangkat lunak komputer penghubung (connectivity software) yang memperbolehkan beberapa buah proses untuk berjalan pada satu atau beberapa buah komputer sekaligus pada jaringan computer



7 Standar Middleware

- Object Request Broker (OBR)
- Distributed Computing Environment (DCE)
- Database Access System (DAC)
- Transaction Processing Monitoring (TPM)
- Remote Procedure Call (RPC)
- Message Passing (MP)
- Enterprise Service Bus (ESB)

Web Service &
Service Oriented Architecture
(SOA)

Domain Integrasi Middleware

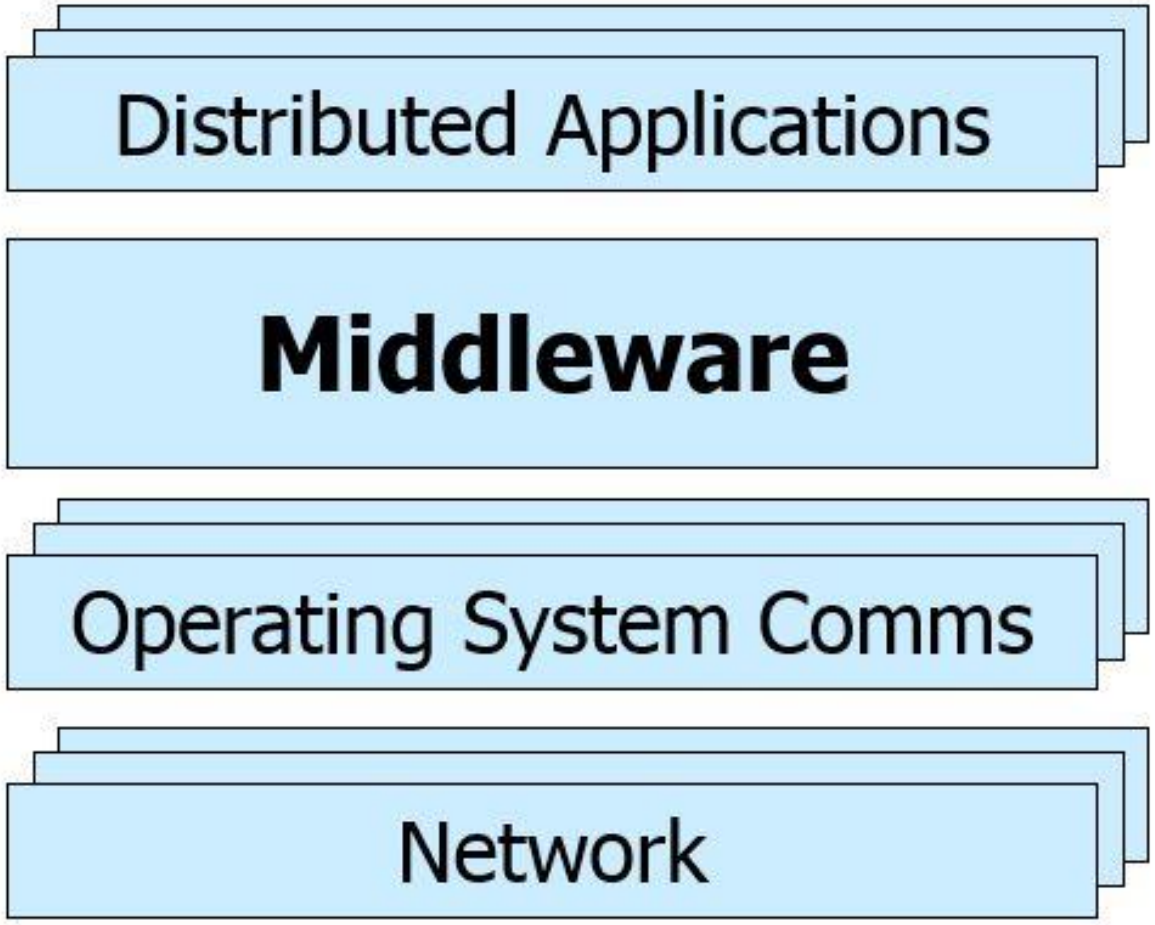
Integrasi Level Middleware membutuhkan sumber daya (resources) komputasi & SDM. Agar proses integrasi berjalan baik integrasi level middleware dipetakan menjadi 4 domain

1. Cloud Integration
   - Cloud Computing mencakup : IAAS, PAAS, SAAS
   - Web service
2. Application Integration
   - Application To Application (A2A) memanfaatkan SAAS Cloud
   - Remote application (protocol : HTTP, FTP, SSH dan Web Service)
3. Database Integration
   - Database, file bisnis, aplikasi business intelligence, aplikasi datawarehouse
4. Business Integration (B2B)

Middleware Technology

Sirojul Munir | rojulman@nurulfikri.ac.id

Lapisan Infrastruktur Aplikasi

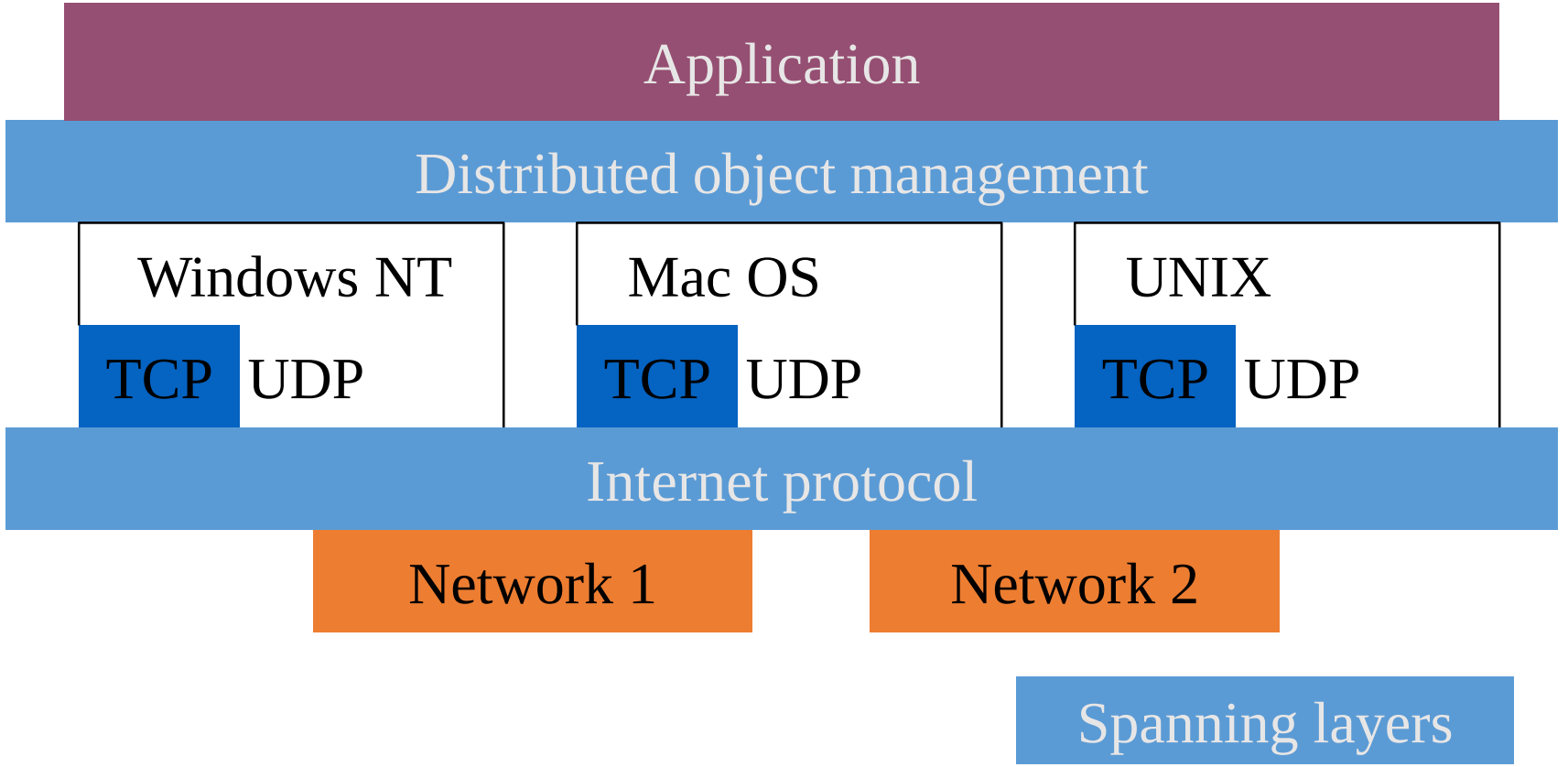


(remote calls, object invocation, messages, …)

(sockets, IP, TCP, UDP, …)

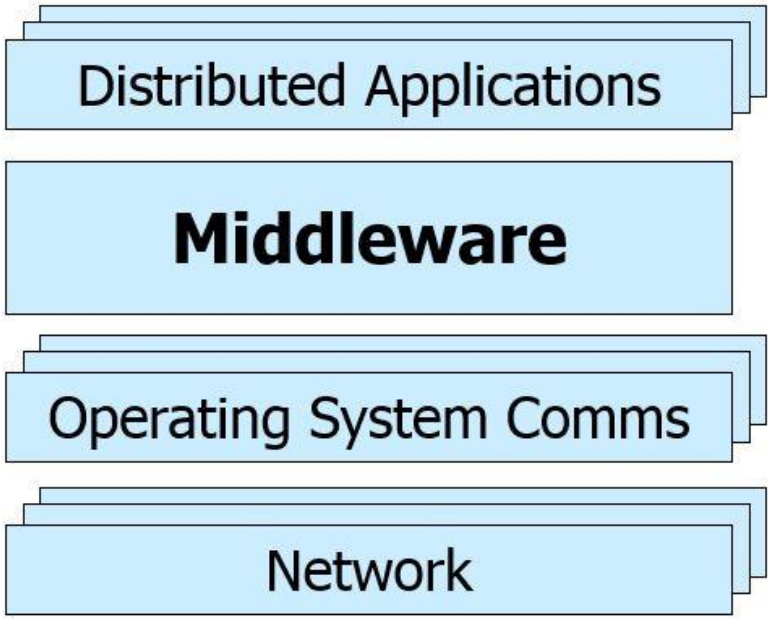
(packets, bits…)

Spanning Layers



Middleware ?

- Lapisan antara aplikasi terdistribusi dengan sistem operasi
- Menyembunyikan kompleksitas dan ke-heterogenan dari sistem terdistribusi
- Jembatan penghubung antara low-level OS dengan abstraksi bahasa pemrograman
- *Menyediakan common programming abstraction (services) and infrastructure* untuk sistem terdistribusi



Middleware Objectives ?

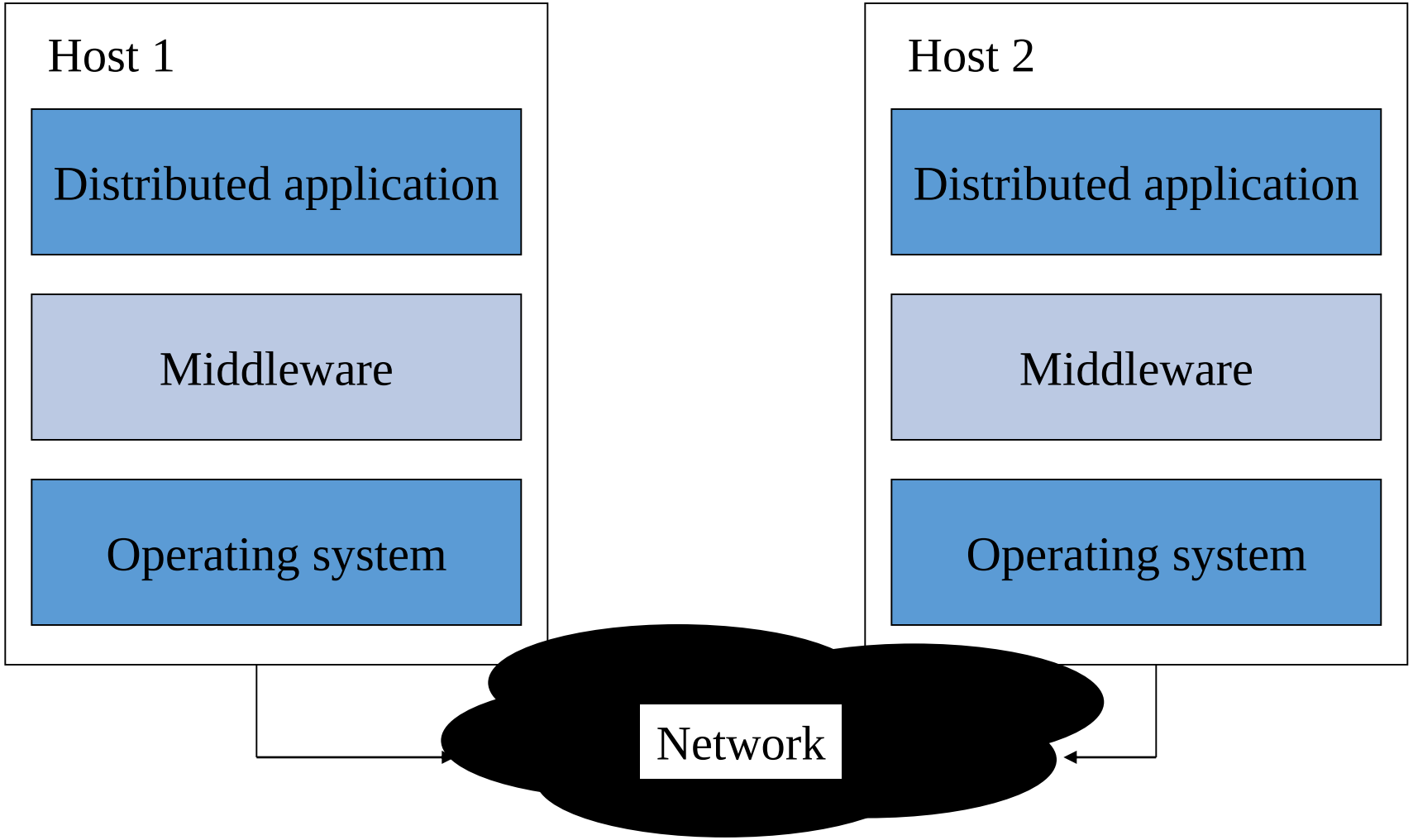
- Menyembunyikan ke-heterogenan
- Indepensi lokasi (location independence)
- Menyediakan fungsionalitas sejenis yang dibutuhkan oleh banyak aplikasi
- Software yang portable dan mobile
- Membantu dalam integrasi aplikasi yang ada (legacy facilities)
- Alat untuk *aplication interoperabilty (apa itu interoperability ?)*
- Alat untuk *scability (apa itu scability ?)*

Middleware Area Support & Dimensions

- Middleware provides support for (some of):
  - Naming, Location, Service discovery, Replication
  - Protocol handling, Communication faults, QoS
  - Synchronisation, Concurrency, Transactions, Storage
  - Access control, Authentication

- Middleware dimensions:
  - Request/Reply vs. Asynchronous Messaging
  - Language-specific vs. Language-independent
  - Proprietary vs. Standards-based
  - Small-scale vs. Large-scale
  - Tightly-coupled vs. Loosely-coupled components

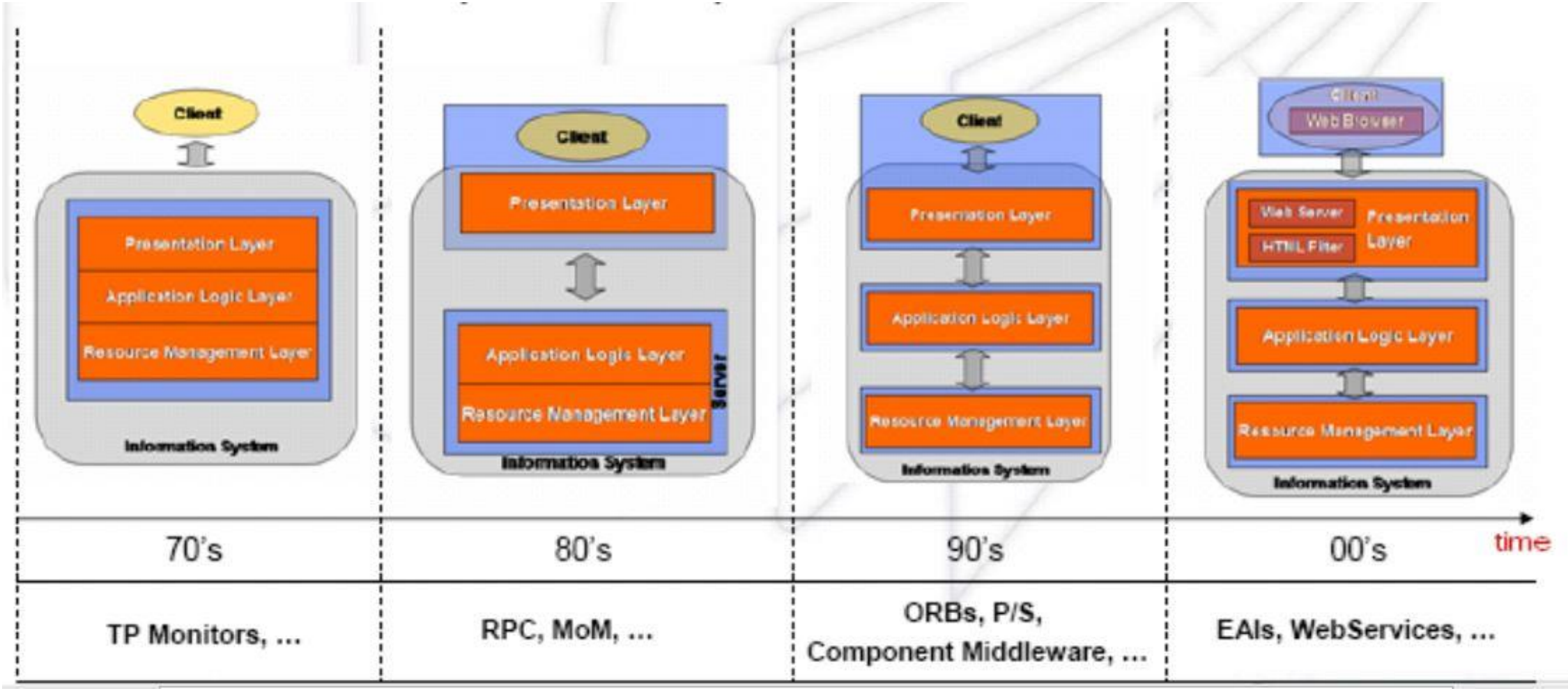
Middleware – Network System



Middleware Technology

- Transaction processing
  - Menyederhanakan koordinasi antara *complementary resource manager*
- Message-oriented middleware
  - Mendukung kemampuan melakukan pengiriman pesan dan antrian ketika *resource manager* tidak tersedia secara simultan (mirip sistem workflow)
- Distributed object management
  - Mendukung aplikasi yang terdistribusi diantara ke-heterogenan platforms dan organisasi
- Mobile code
  - Memungkinkan kode aplikasi dapat dipindahkan dan di eksekusi pada platform yang heterogen
  - Tidak membutuhkan instalasi software sebelumnya

Evolusi Middleware



Middleware :: Services Category

1. **Data Management Services**
   Database and file system middleware
2. **Communication Services**
   RPC (Remote Procedure Call) and **messaging** middleware
3. **Distribution Services**
   location, time and security services
4. **Object Management Services**
   using Object Request Brokers (**ORBs**).

Middleware :: Services Category

5. **Application Co-operation Services**

   Transaction-Processing (**TP**) monitors, e-mail, etc
6. **Presentation Services**

   User Interfaces, printing and multi-media middleware
7. **System Management Services**

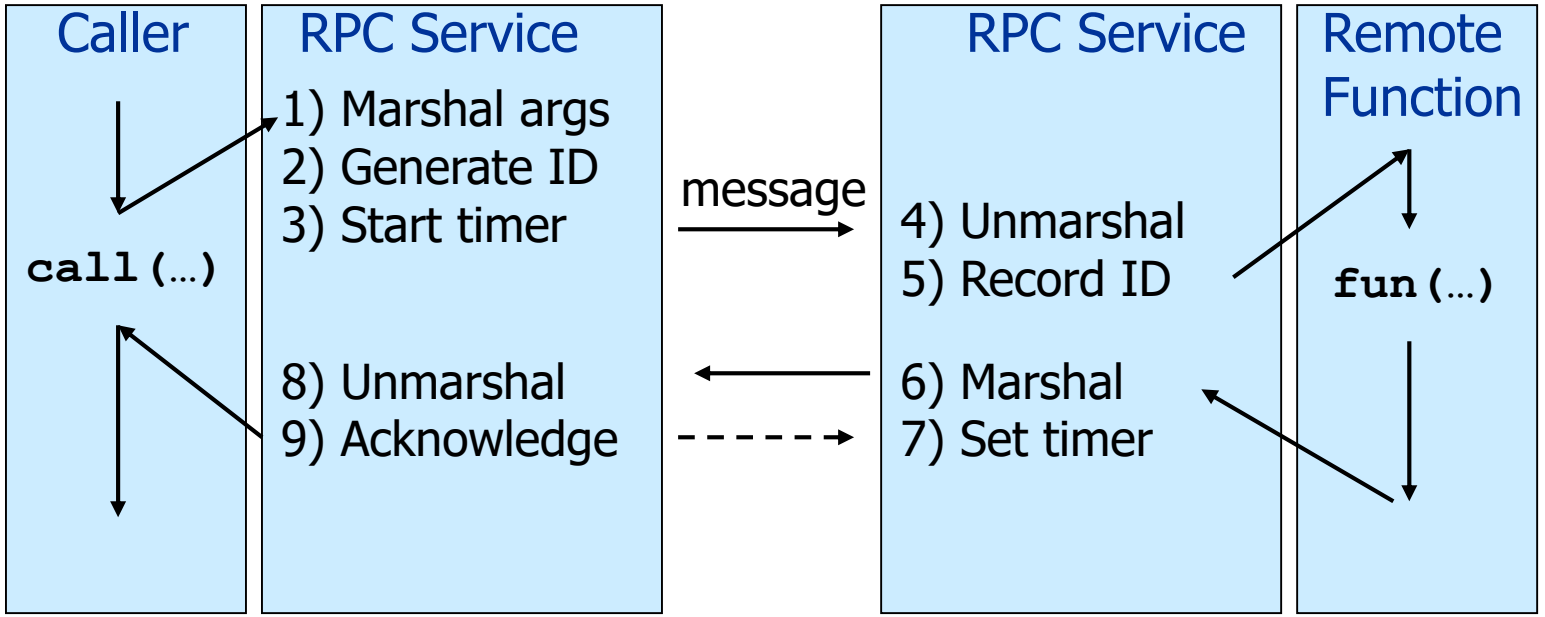
   Configuration-, change-, operations-, problem-, and performance-management services

Data Management Services

- Aplikasi yang menyediakan layanan akses ke database melalui API Database Seperti ODBC , JDBC , SQL Services
- Aplikasi medukung multiplatform database : Oracle, Informix, PostgreSQL, MySQL, SQL Server dll
- Contoh: CI-Link, AccessWorks, Bucado Replication System , JINI

Communication Service : Remote Procedure Call (RPC)

- Masks remote function calls as being local
- Client/server model
- Request/reply paradigm usually implemented with message passing in RPC service
- Marshalling of function parameters and return value



Properties of RPC

Language-level pattern of **function call**

- easy to understand for programmer

Synchronous request/reply interaction

- natural from a programming language point-of-view
- matches replies to requests
- built in synchronisation of requests and replies

Distribution transparency (in the no-failure case)

- hides the complexity of a distributed system

Various **reliability** guarantees

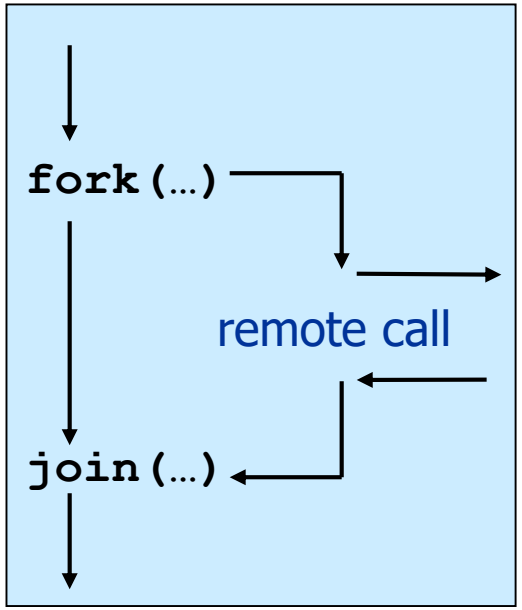
- deals with some distributed systems aspects of failure

Disadvantages of RPC

✖Synchronous request/reply interaction
- tight coupling between client and server
- client may block for a long time if server loaded
  leads to multi-threaded programming at client
- slow/failed clients may delay servers when replying
  multi-threading essential at servers

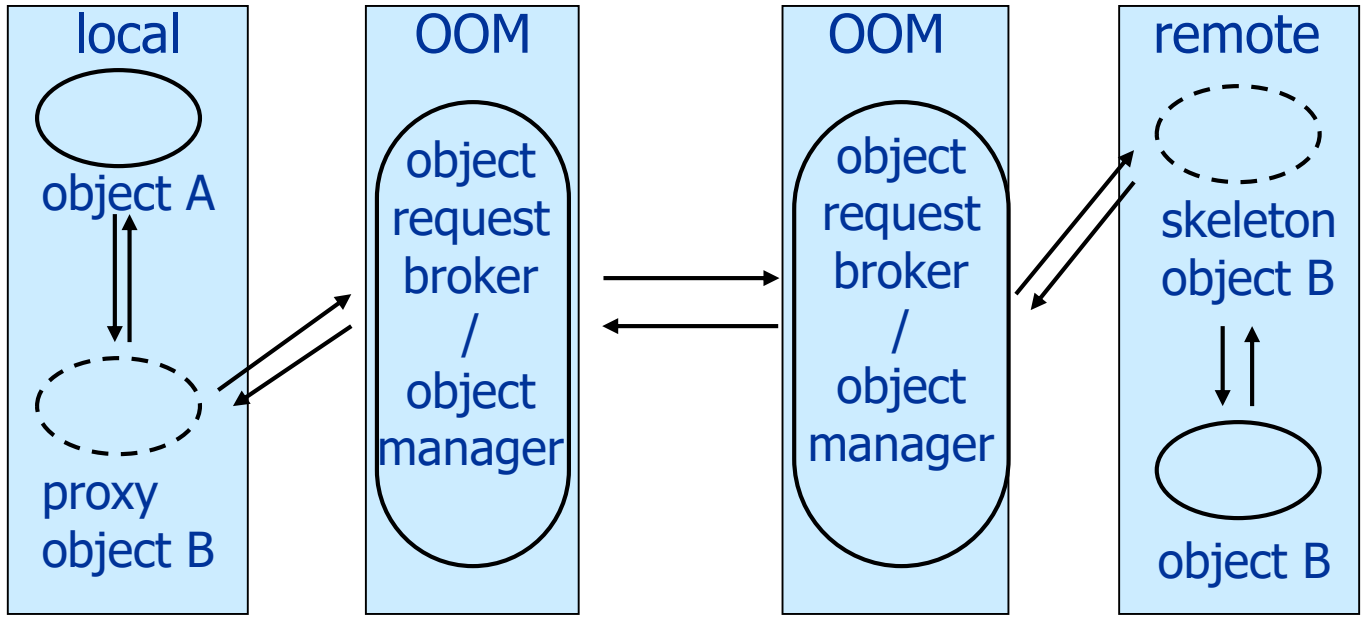
✖Distribution Transparency
- Not possible to mask all problems

✖RPC paradigm is not object-oriented
- invoke functions on servers as opposed to methods on objects



Communication Services: OO Middleware (OOM)

- **Objects** can be *local* or *remote*
- **Object references** can be *local* or *remote*
- Remote objects have visible **remote interfaces**
- Masks remote objects as being local using **proxy objects**
- **Remote method invocation**



Properties of OOM

Support for object-oriented programming model
- objects, methods, interfaces, encapsulation, …
- exceptions (were also in some RPC systems e.g. Mayflower)

Synchronous request/reply interaction
- same as RPC

Location Transparency
- system (ORB) maps object references to locations

Services comprising multiple servers are easier to build with OOM
- RPC programming is in terms of ***server-interface (operation)***
- RPC system looks up server address in a location service

Java Remote Method Invocation (RMI)

- Covered in 1B Advanced Java programming
- Distributed objects in Java

```
public interface PrintService extends Remote {
   int print(Vector printJob) throws RemoteException;
}
```

- RMI compiler creates proxies and skeletons
- RMI registry used for interface lookup
- Entire system written in Java (single-language system)