SOFTWARE ARCHITECTURES

ARCHITECTURAL STYLES

SCALING UP PERFORMANCE

ARCHITECTURES

SW Architectures usually complex

Often we reduce the abstraction

Architectural Styles

- Layered style

Architectural Patterns

- Model View Controller

ARCHITECTURE STYLES

Basic Characteristics

Quality attributes

ARCHITECTURE STYLES

Data centric

- Databases

Call and return

- Part of this course

Implicit invocation

- Events

Independent components

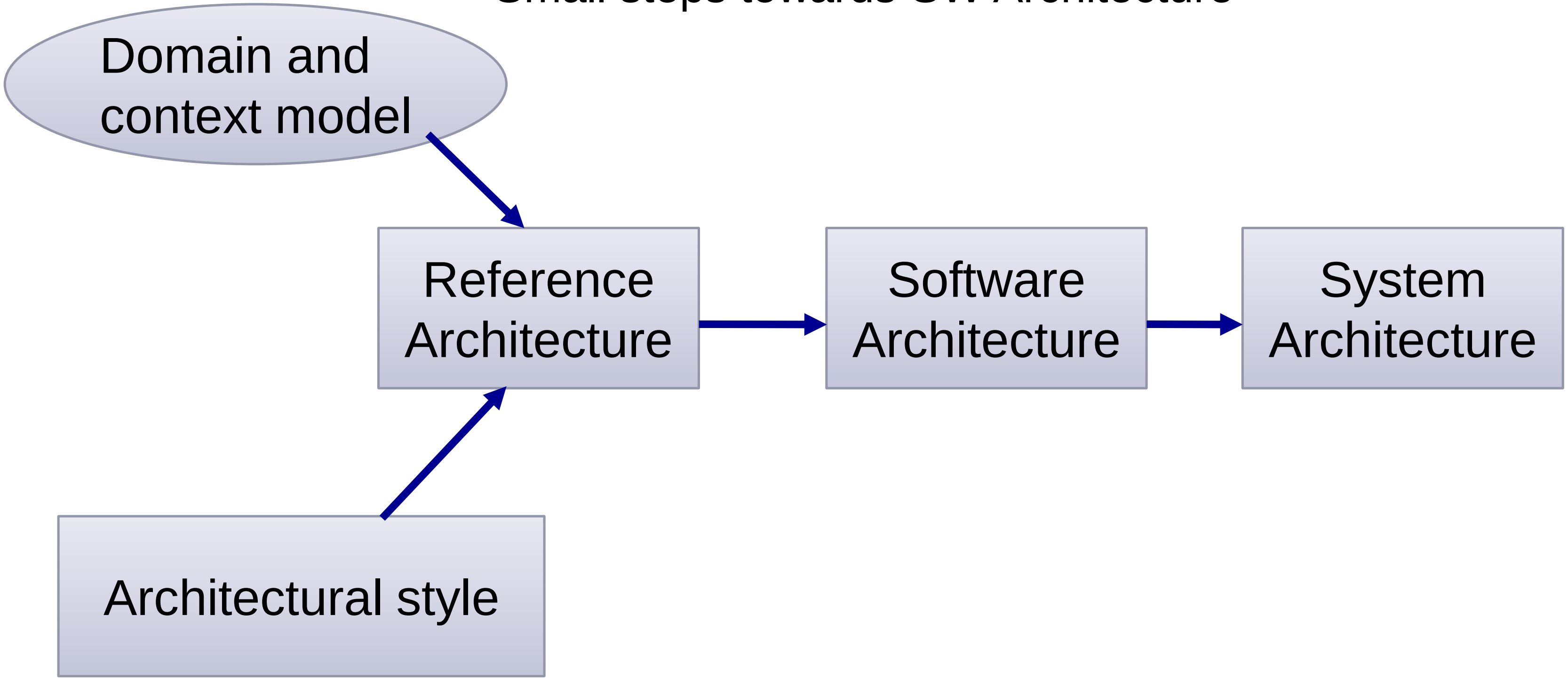
- Peer to peer

Virtual Machines

Pipe and Filter - data flow

OVERVIEW

- **Domain and context model**
- **Arch. styles**
- **Reference architecture**
  - Small steps towards SW Architecture


Domain and context model

Reference Architecture

Software Architecture

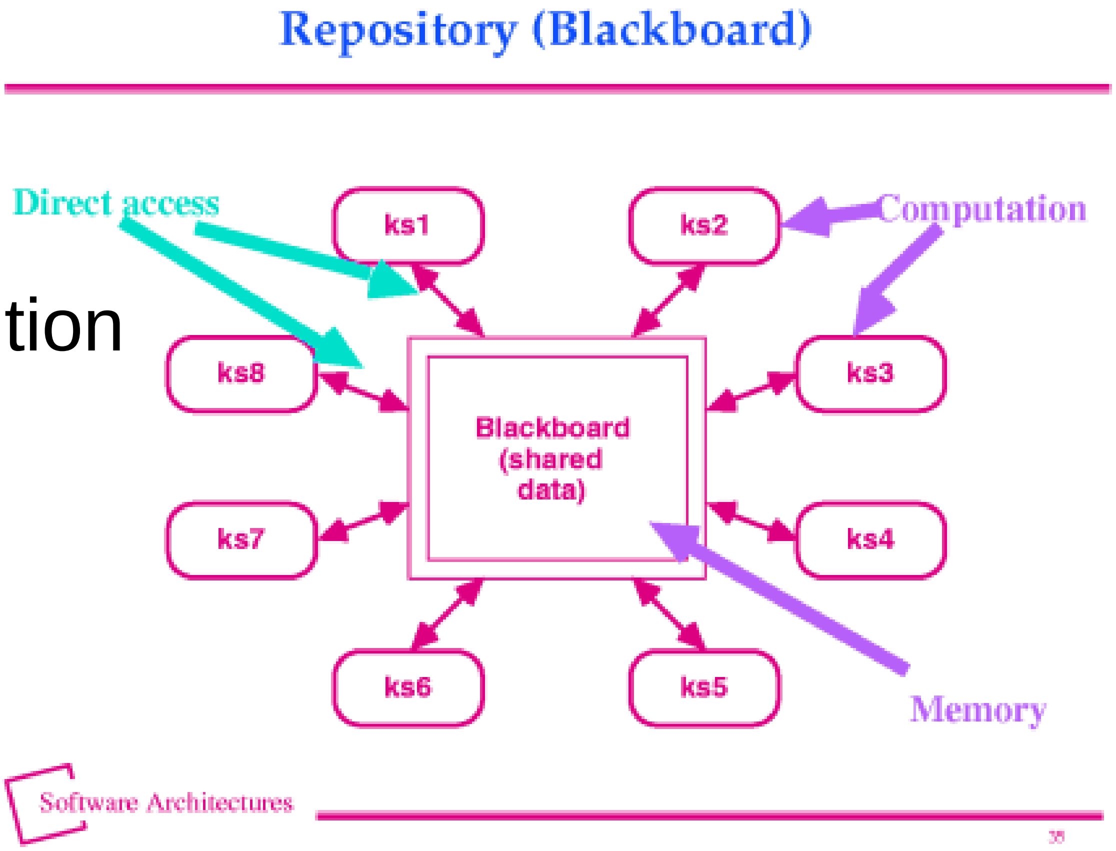
System Architecture

Architectural style

ARCHITECTURE STYLES

Data centric

- Databases
- Voice recognition
- Compilers

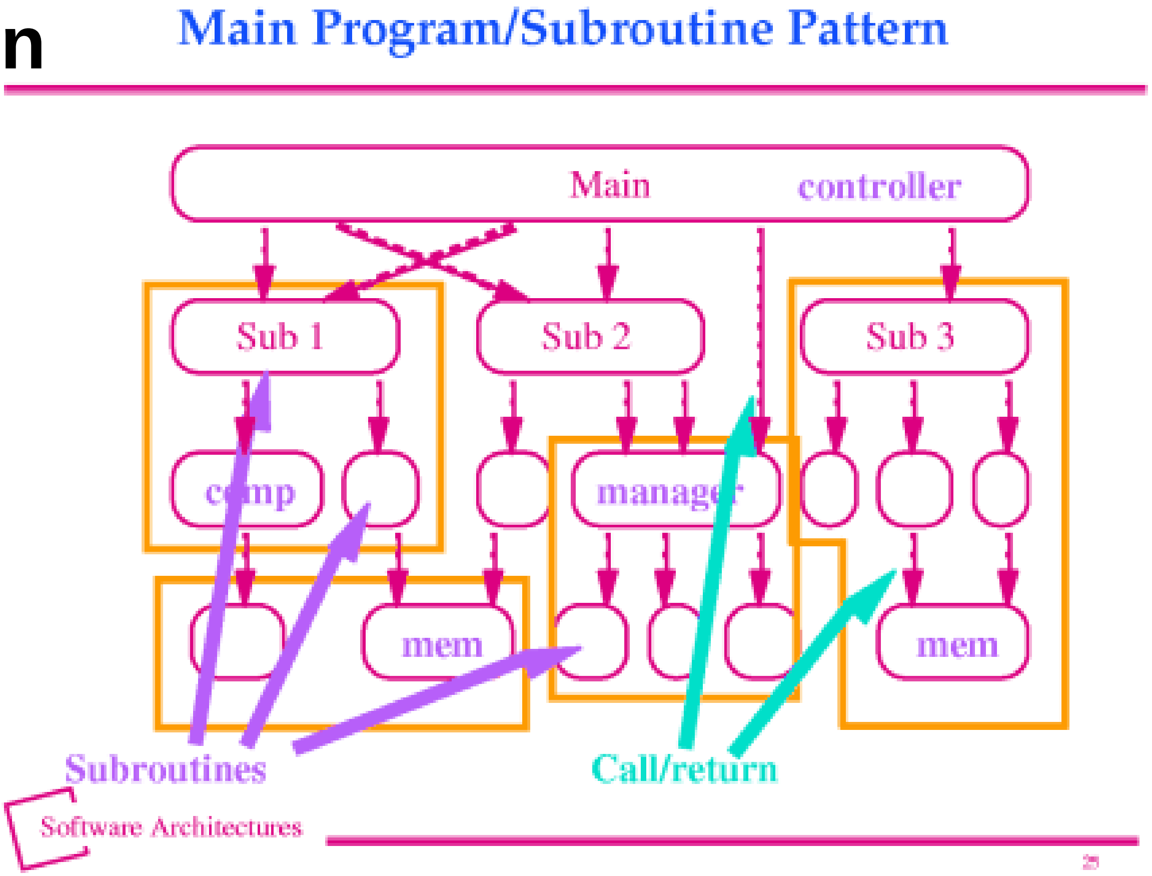


Repository (Blackboard)

ARCHITECTURE STYLES

Call and return

- OOD
- Procedural
- RPC
- AOP
- Layers

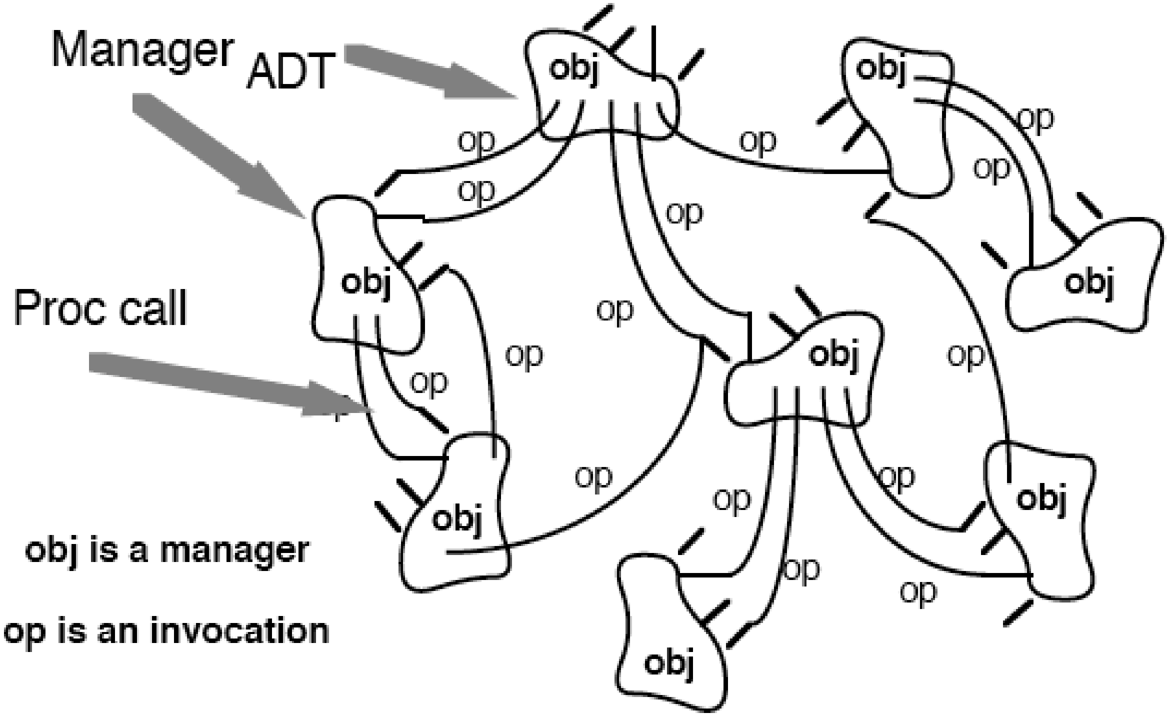


Main Program/Subroutine Pattern

ARCHITECTURE STYLES

Call and return

- OOD
- Procedural
- RPC
- AOP
- Layers



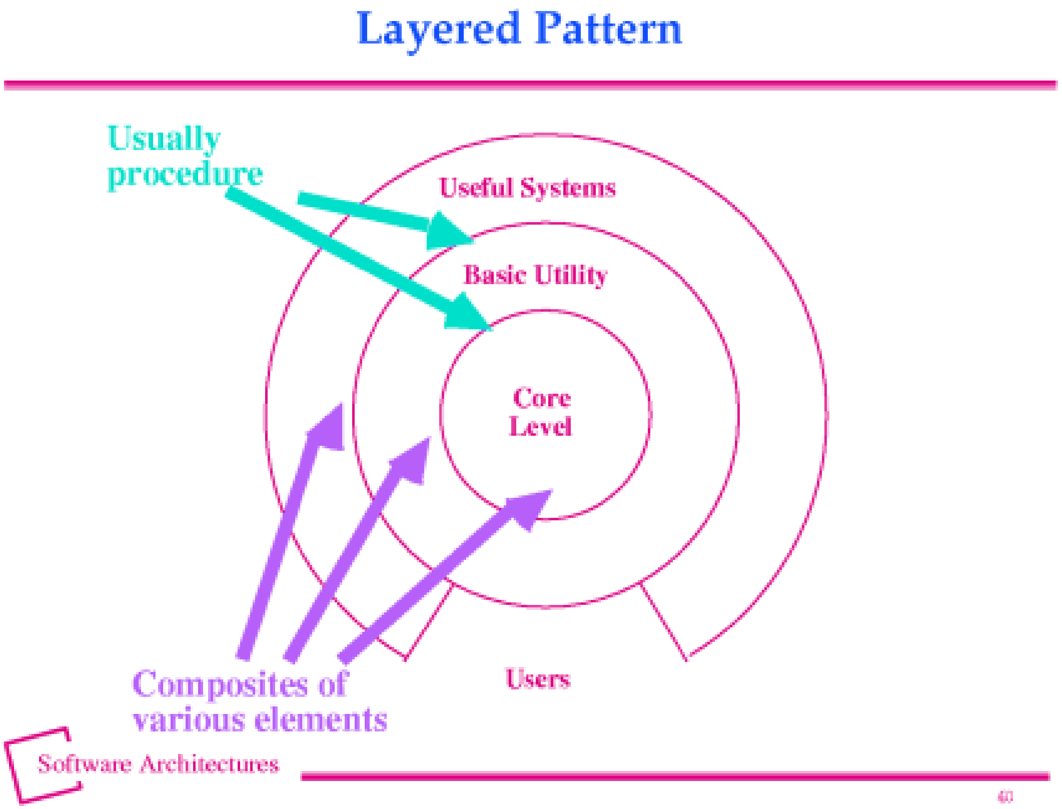
obj is a manager

op is an invocation

ARCHITECTURE STYLES

Call and return

- OOD
- Procedural
- RPC
- AOP
- Layers

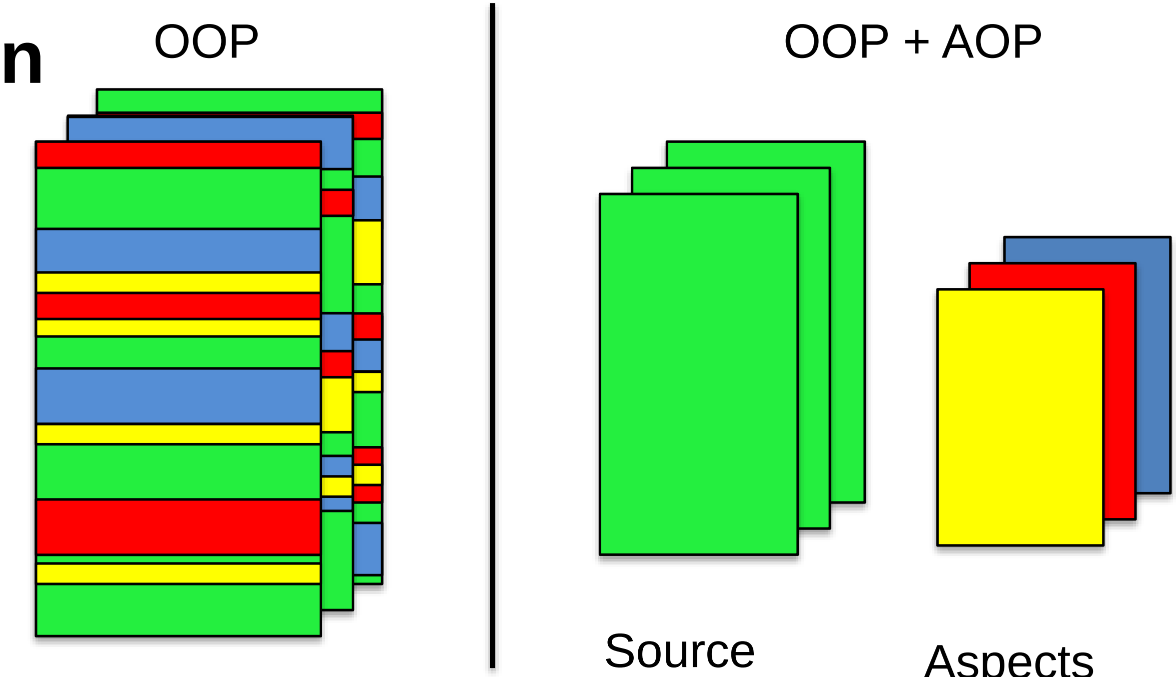


Layered Pattern

ARCHITECTURE STYLES

Call and return

- OOD
- Procedural
- RPC
- AOP
- Layers



OOP

OOP + AOP

Source code of methods

Source code of methods

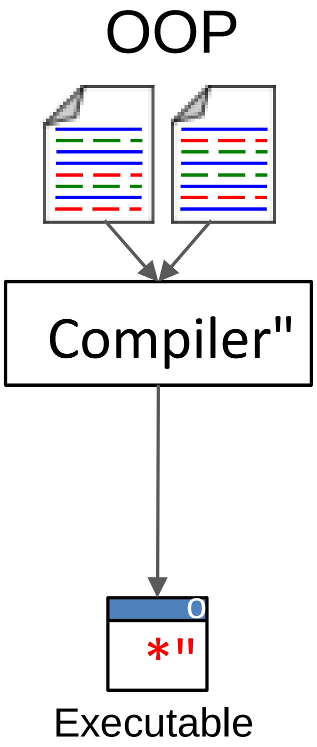
Aspects

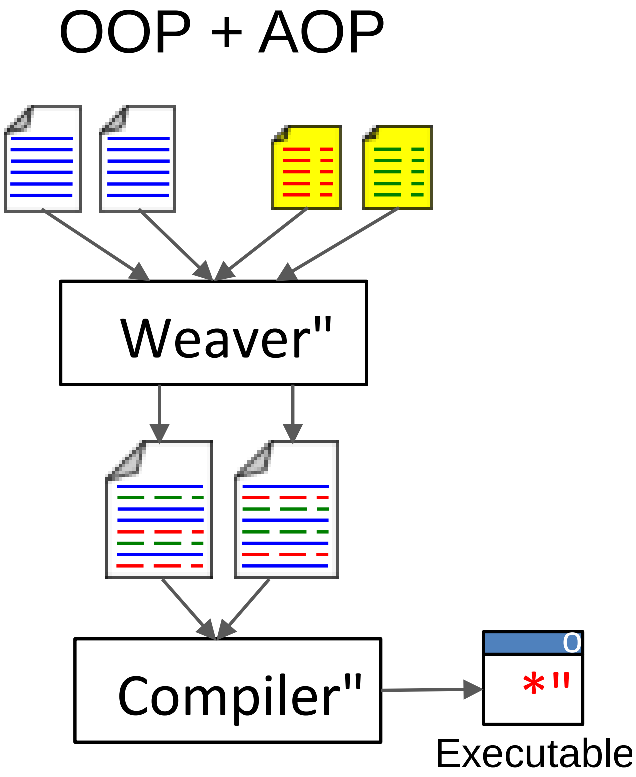
🟥 Security 🟩 Method logic 🟦 Synchronization 🟨 Logging

ARCHITECTURE STYLES

Call and return

- OOD
- Procedural
- RPC
- AOP
- Layers

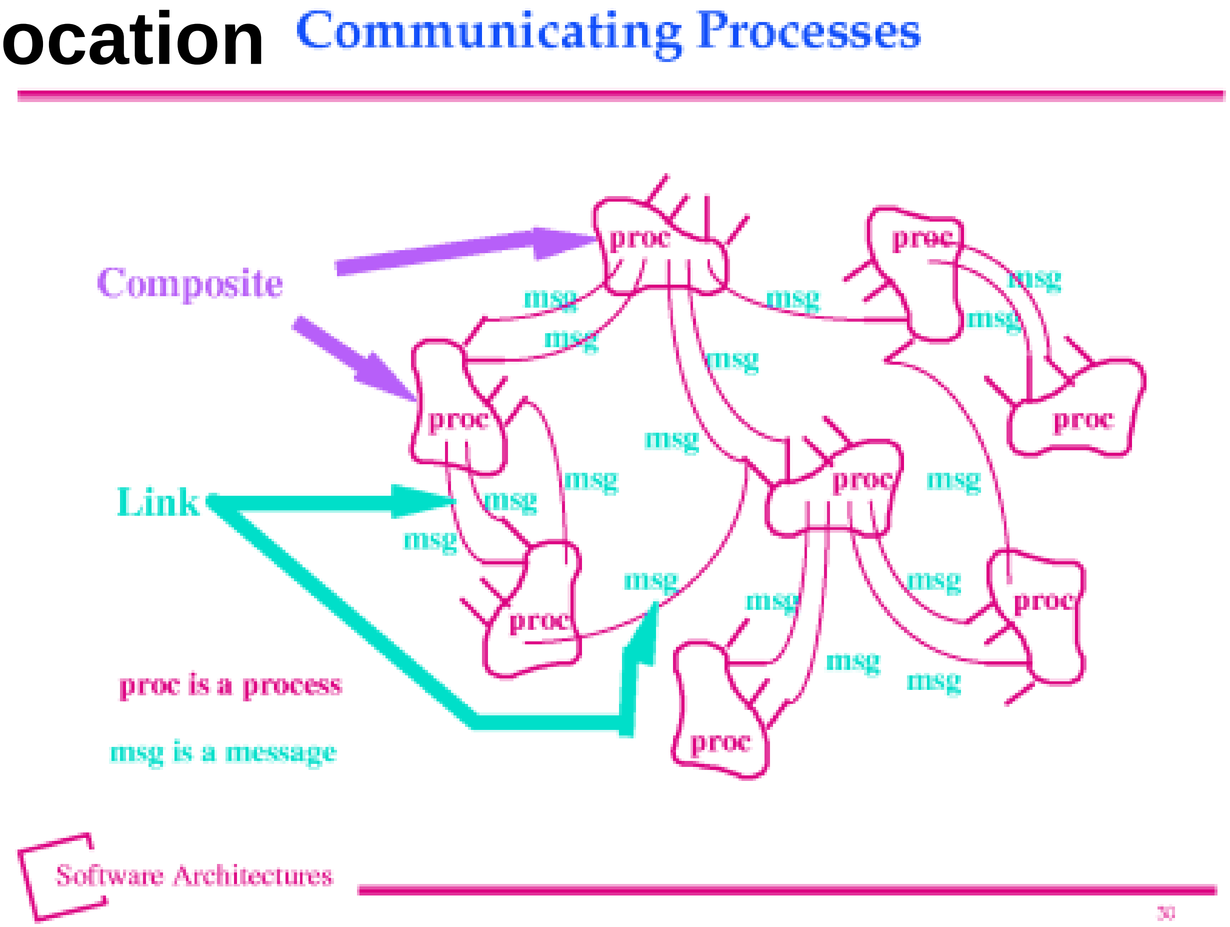




ARCHITECTURE STYLES

Implicit invocation

- Events



Communicating Processes

proc is a process

msg is a message

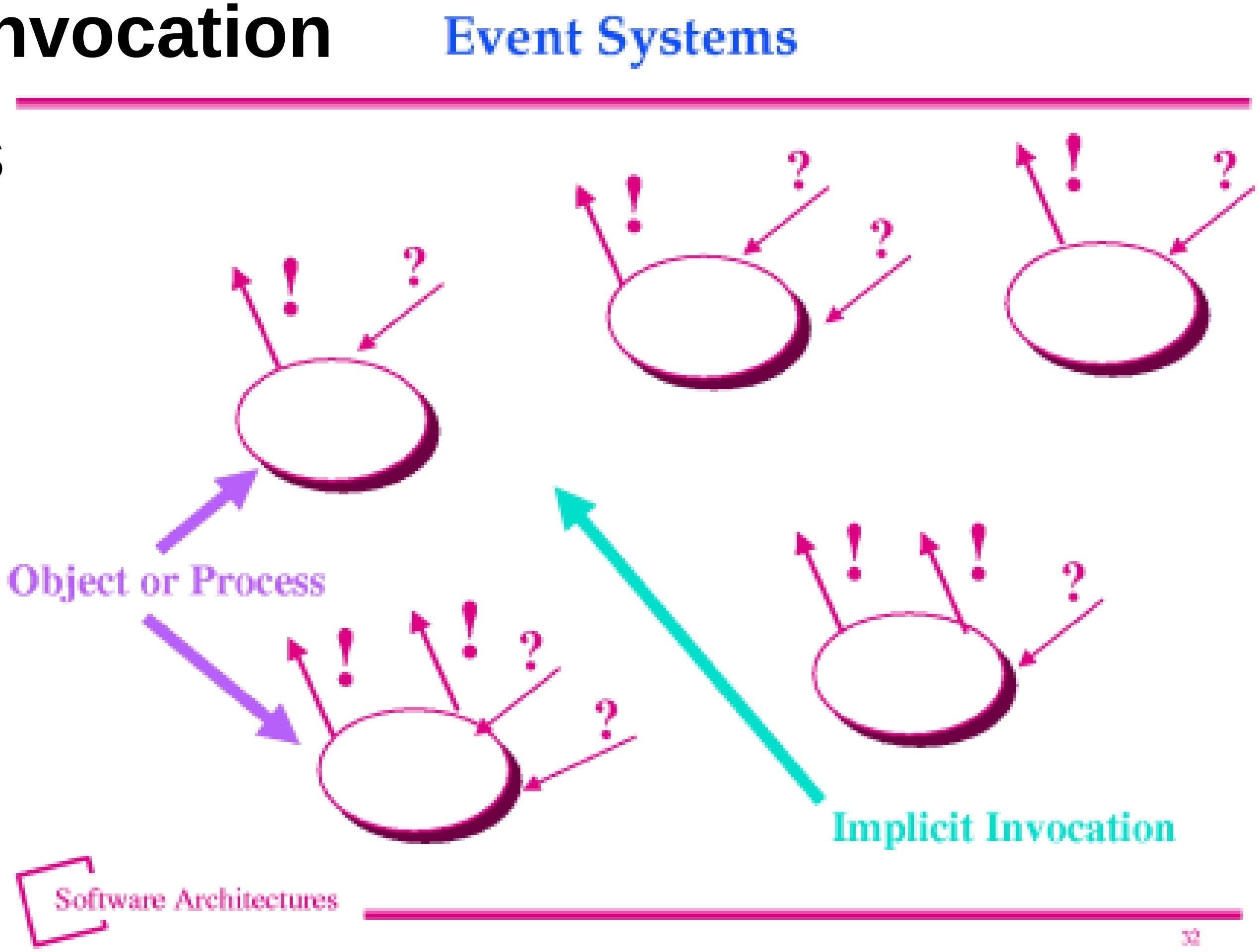
Software Architectures

30

ARCHITECTURE STYLES

Implicit invocation

- Events



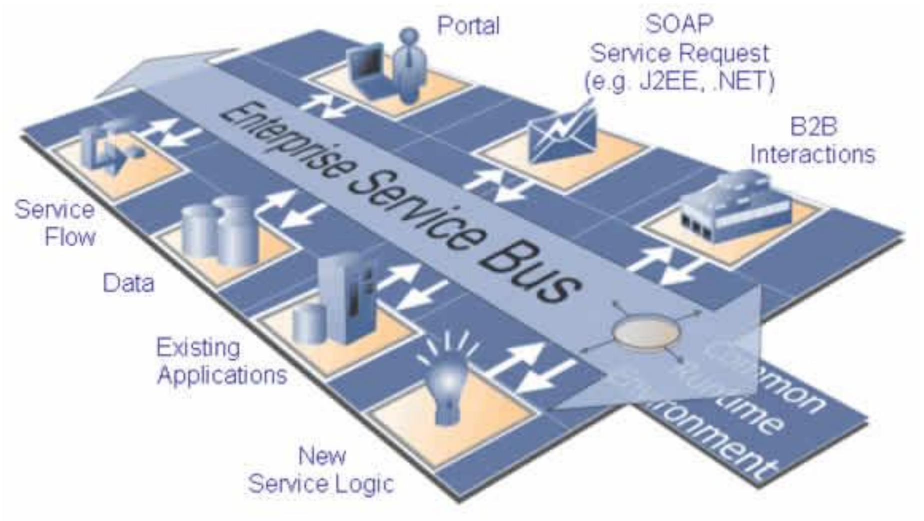
Event Systems

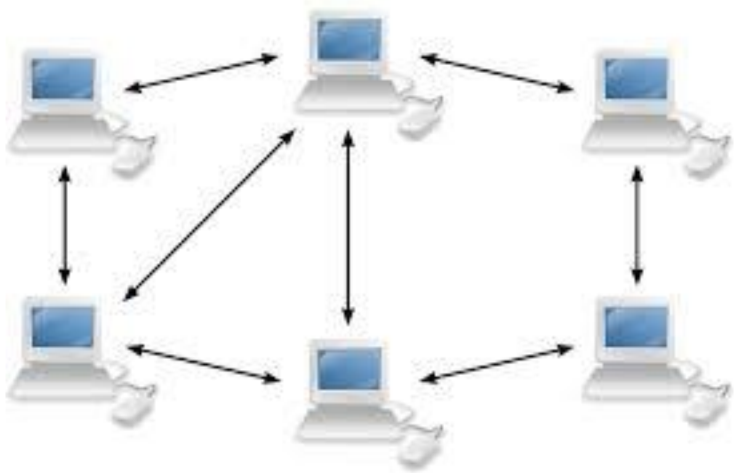
Software Architectures

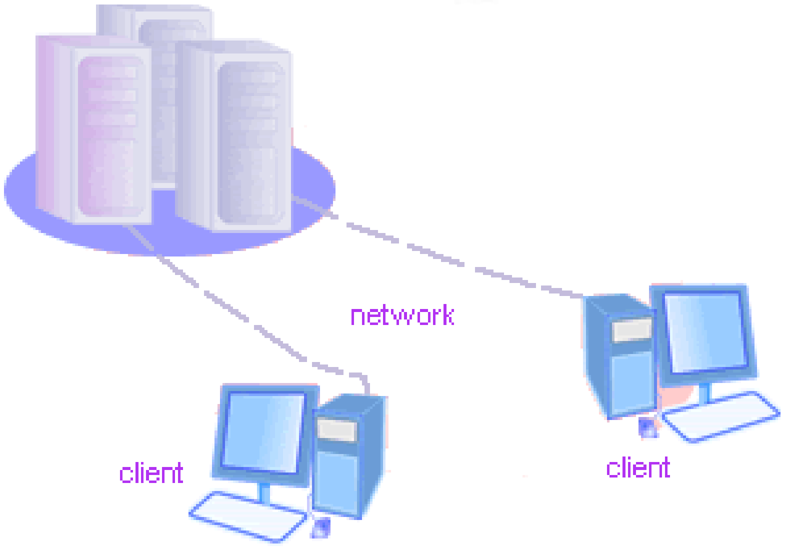
32

ARCHITECTURE STYLES

Independent components

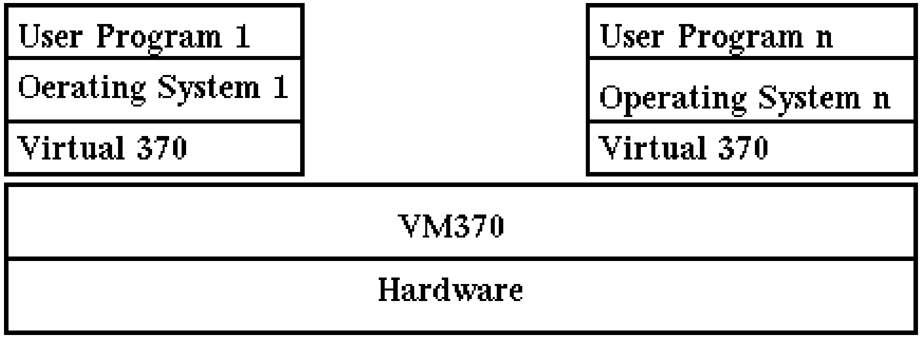




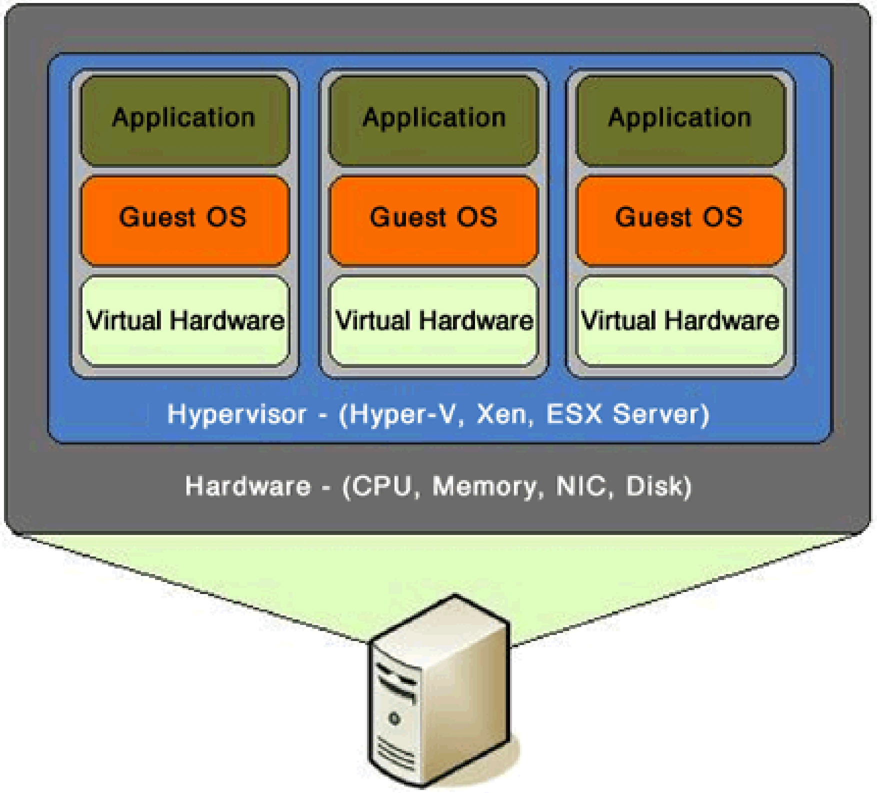


ARCHITECTURE STYLES

Virtual machines

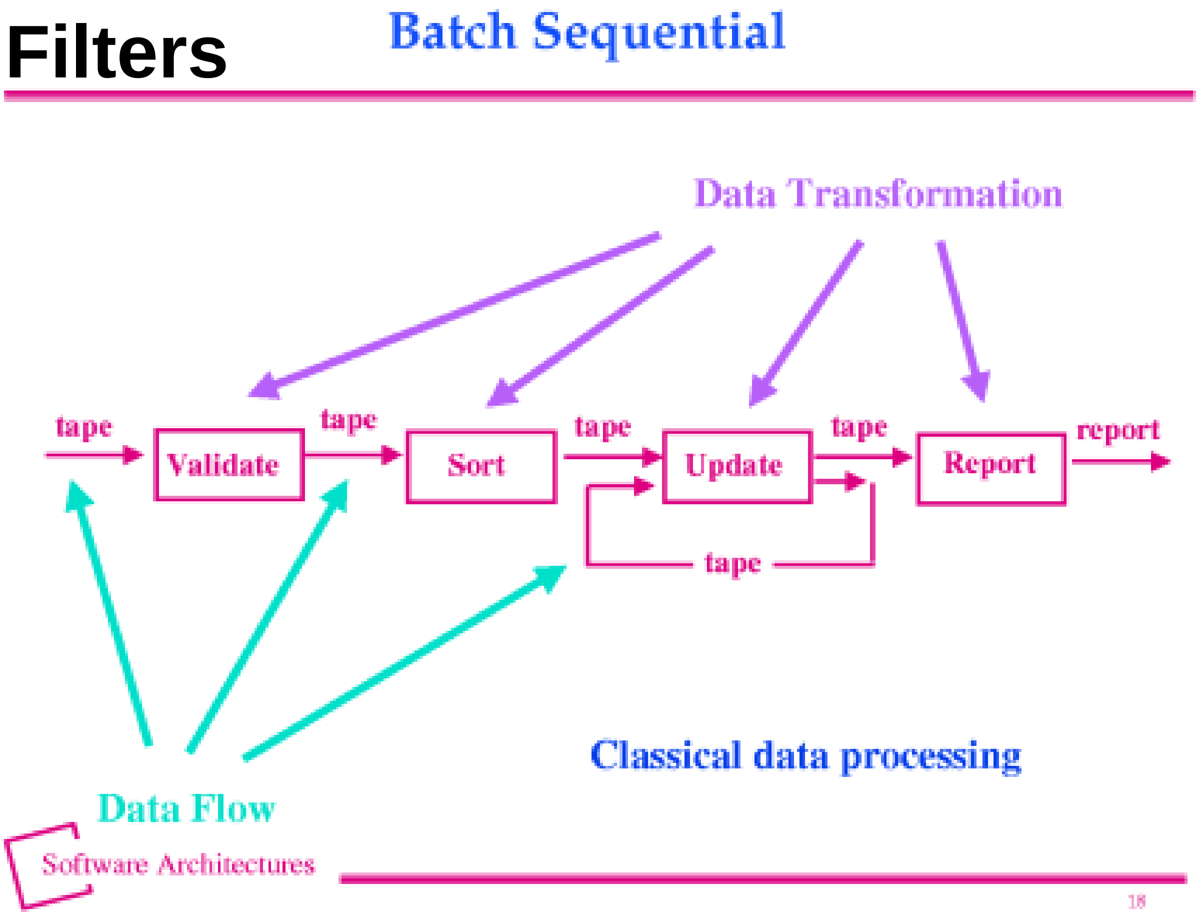


VIRTUAL MACHINE ARCHITECTURE (VM370)



ARCHITECTURE STYLES

Pipes and Filters



Batch Sequential

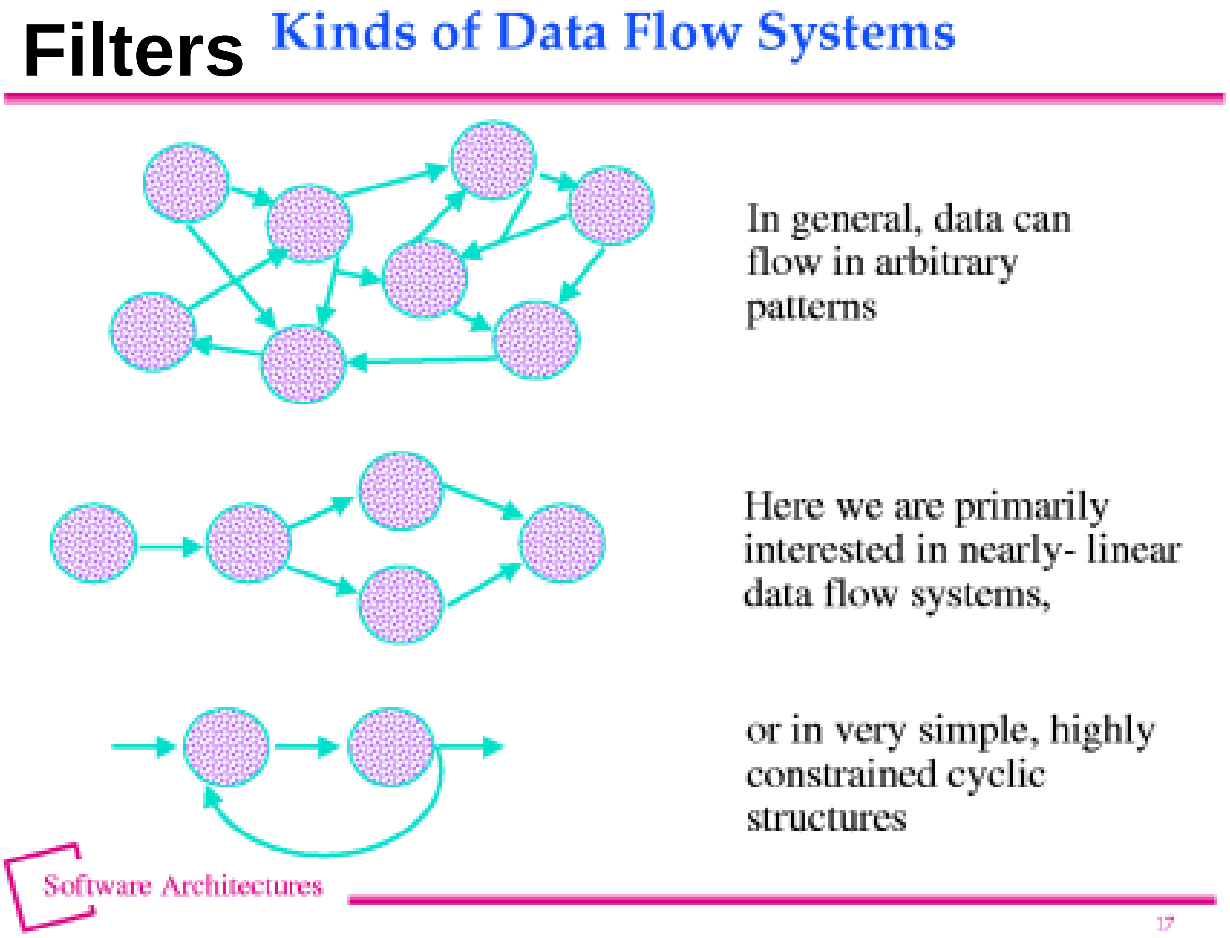
Classical data processing

Software Architectures

18

ARCHITECTURE STYLES

Pipes and Filters



Kinds of Data Flow Systems

In general, data can flow in arbitrary patterns

Here we are primarily interested in nearly- linear data flow systems,

or in very simple, highly constrained cyclic structures

Software Architectures

17

ARCHITECTURE STYLES

Data centric

- Data integration, Distribution, Control, Coordination
- Scalability, Low coupling, Centralization, Reuse, Modifiable,

Call and return

- Modifiable, Reusable, Inf. hiding, Structural decomposition, Separation of concerns

Implicit invocation

- Modifiable, Low coupling, Hard to comprehend,

Independent components

- Integration, Scalability, Reuse, Low coupling, Distribution, Reliability

Virtual Machines

- Simulation, Emulation, Portability!, Flexibility, Lowered Performance, Extended features

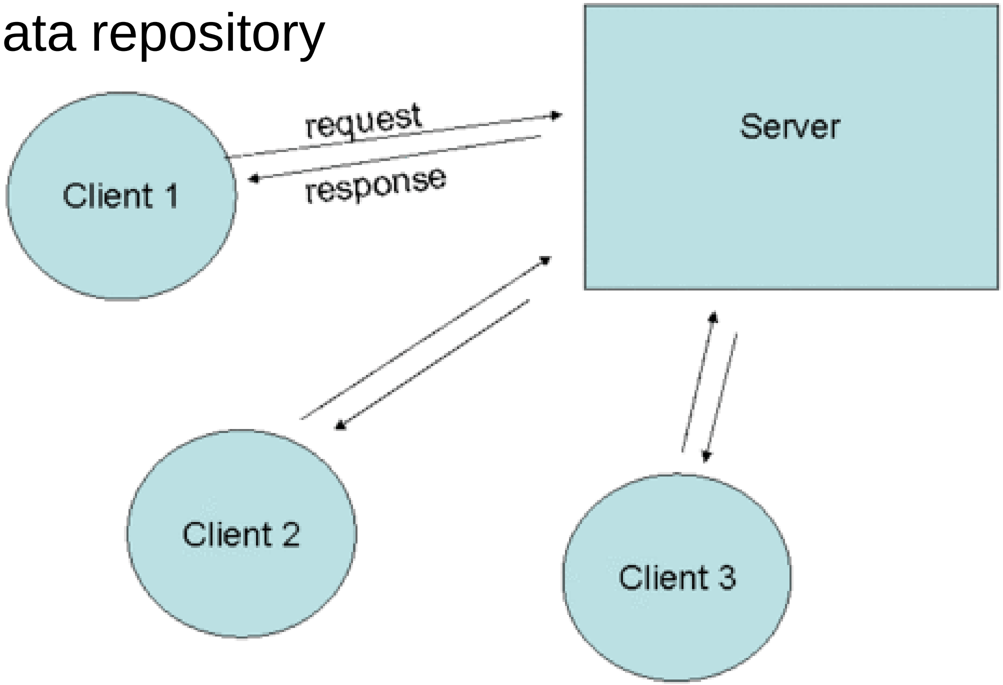
Pipe and Filter

- Modifiable, Reuse, Easy design, Simplicity, Low Coupling,
- Slow, No filter cooperation, Lot of parsing

SCALING PERFORMANCE

Usual approach is to deploy app to a web server and provide access through HTTP/S

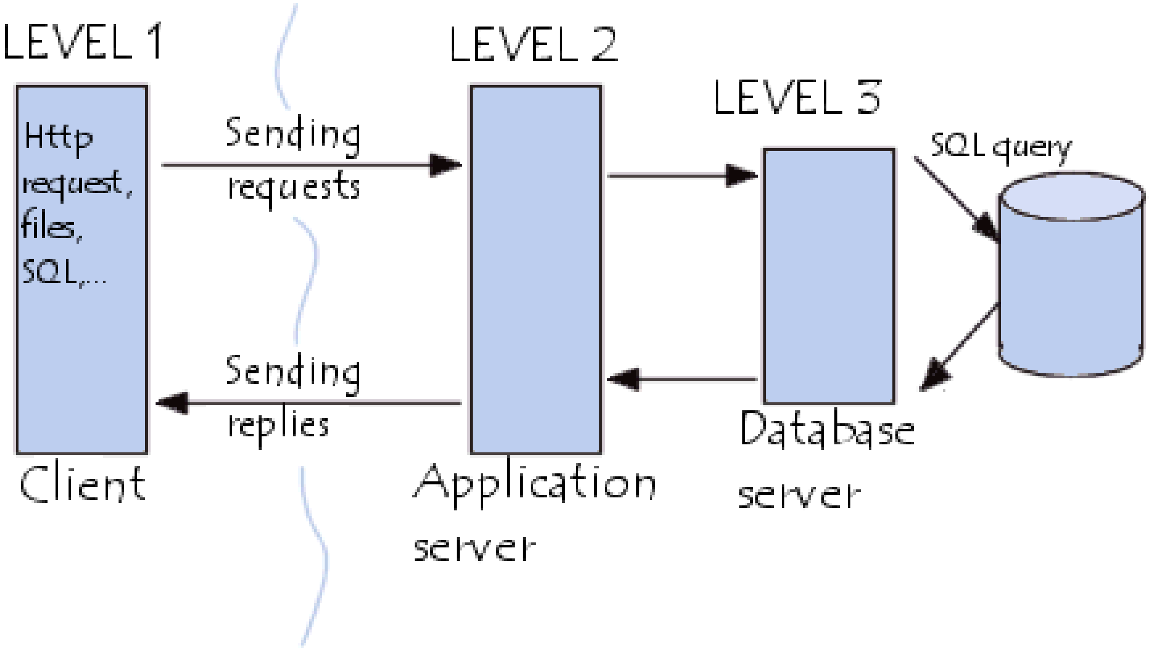
- **Client-server architecture**
  - Inside 3-layers and data repository



SCALING PERFORMANCE

Usual approach is to deploy app to a web server and provide access through HTTP/S

- **Client-server architecture**
  - Inside 3-layers and data repository

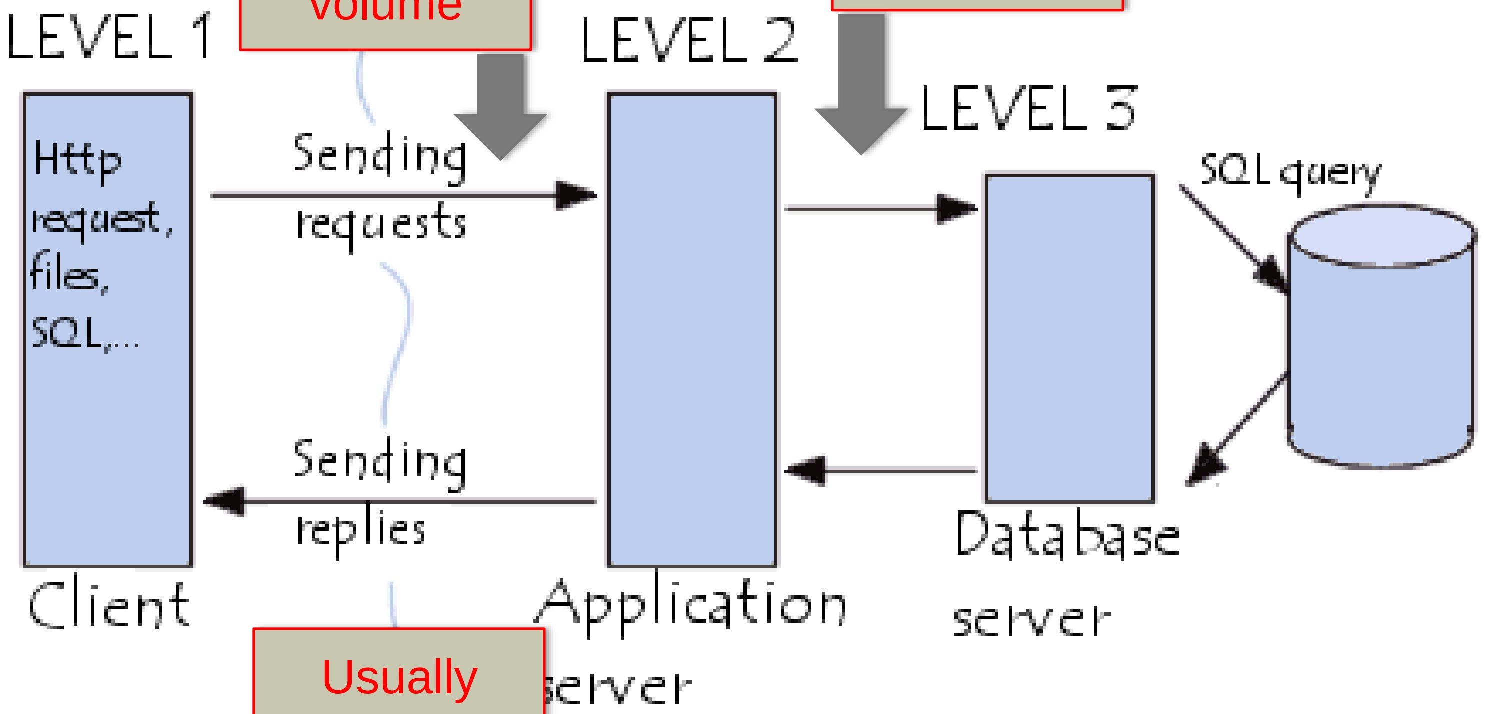


SCALING PERFORMANCE

ORM talks too much when not being careful

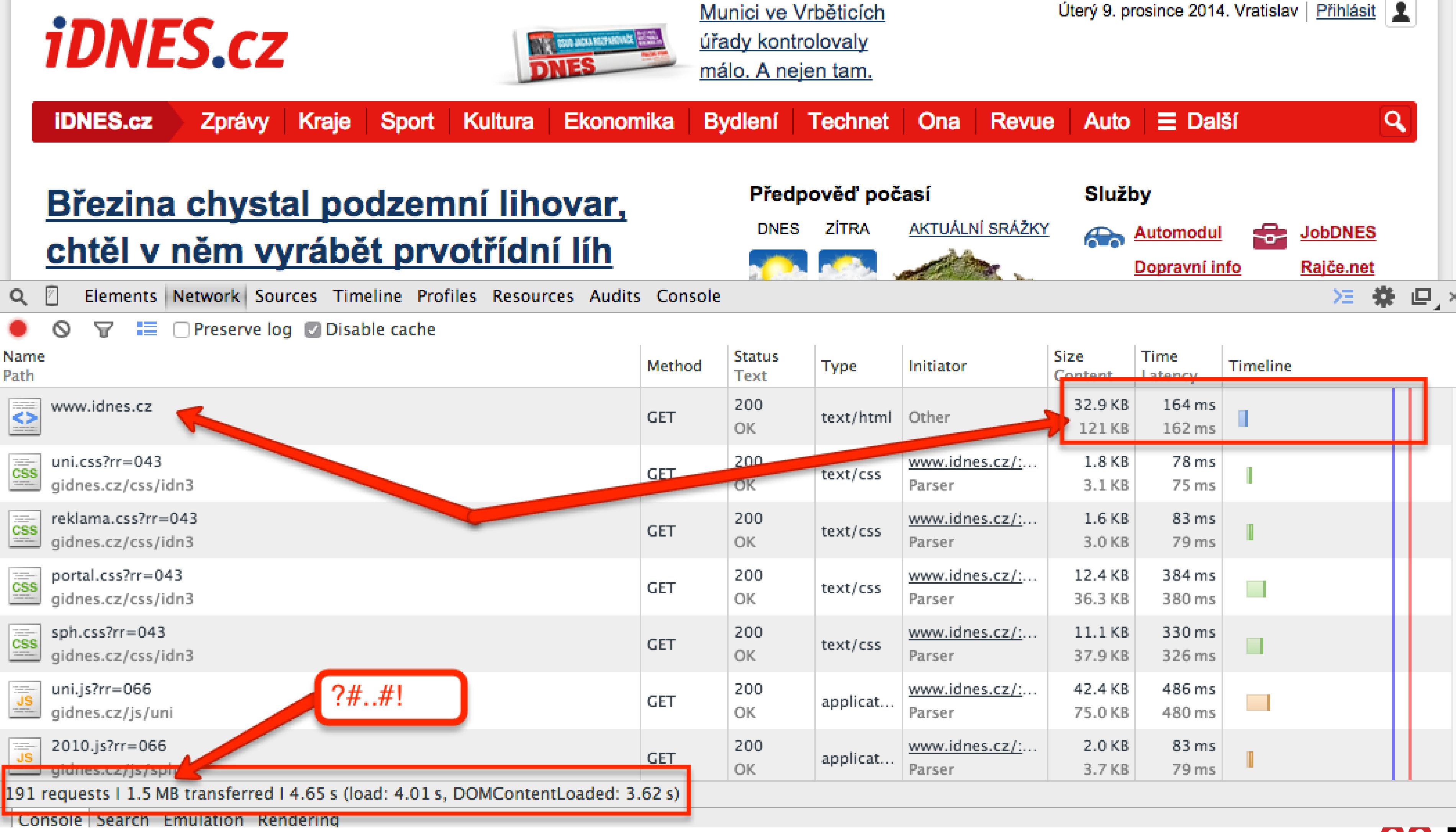
Minimize volume

Minimize volume



Usually HTTP (JS/CSS)

DEPLOYMENT, MAINTENANCE AND REPORTS



iDNES.cz

Munici ve Vrběticích úřady kontrolovaly málo. A nejen tam.

Úterý 9. prosince 2014. Vratislav | Přihlásit

Březina chystal podzemní lihovar, chtěl v něm vyrábět prvotřídní líh

Předpověď počasí

Služby

☐ Preserve log ☑ Disable cache

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
www.idnes.cz	GET	200 OK	text/html	Other	32.9 KB 121 KB	164 ms 162 ms	
uni.css?rr=043 gidnes.cz/css/idn3	GET	200 OK	text/css	www.idnes.cz/:... Parser	1.8 KB 3.1 KB	78 ms 75 ms	
reklama.css?rr=043 gidnes.cz/css/idn3	GET	200 OK	text/css	www.idnes.cz/:... Parser	1.6 KB 3.0 KB	83 ms 79 ms	
portal.css?rr=043 gidnes.cz/css/idn3	GET	200 OK	text/css	www.idnes.cz/:... Parser	12.4 KB 36.3 KB	384 ms 380 ms	
sph.css?rr=043 gidnes.cz/css/idn3	GET	200 OK	text/css	www.idnes.cz/:... Parser	11.1 KB 37.9 KB	330 ms 326 ms	
uni.js?rr=066 gidnes.cz/js/uni	GET	200 OK	applicat...	www.idnes.cz/:... Parser	42.4 KB 75.0 KB	486 ms 480 ms	
2010.js?rr=066 gidnes.cz/js/spl	GET	200 OK	applicat...	www.idnes.cz/:... Parser	2.0 KB 3.7 KB	83 ms 79 ms	

?#..#!

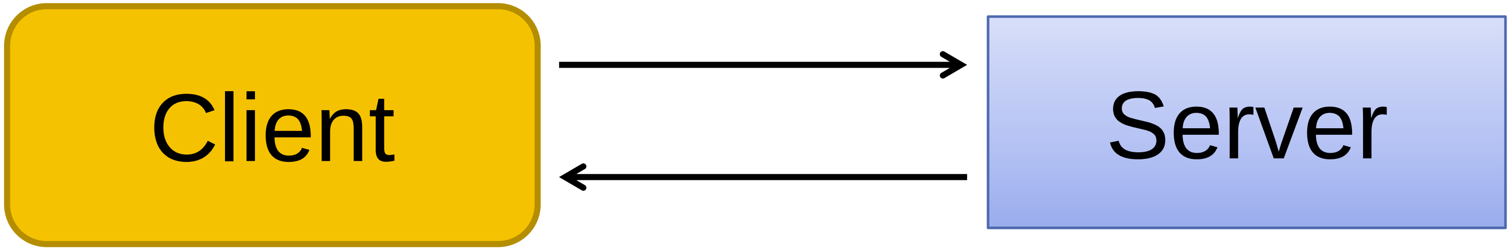
191 requests | 1.5 MB transferred | 4.65 s (load: 4.01 s, DOMContentLoaded: 3.62 s)

CLIENT-SERVER ARCHITECTURE

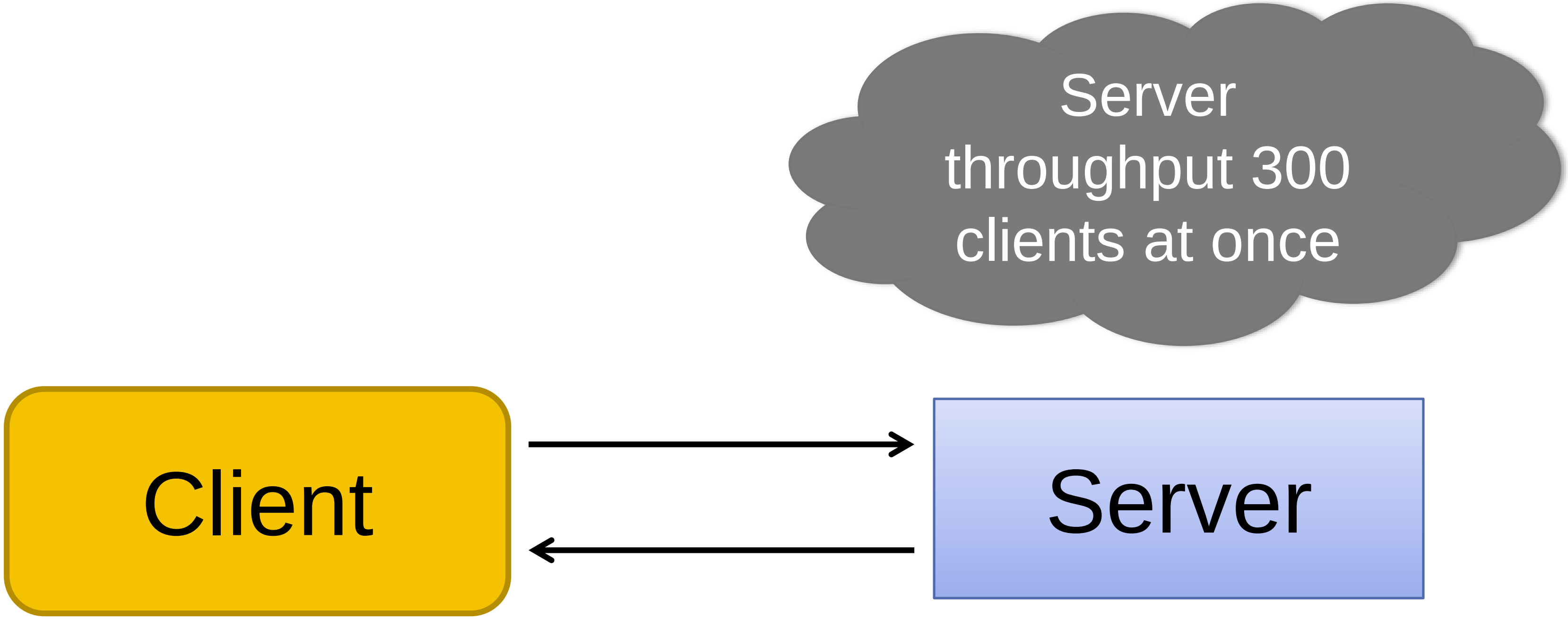
Properties:

- Centralization
- Easy with security
- Easy to locate
- Easy to scale
  - Until we reach the limit
  - Server is the bottleneck
- Performance influenced by the network conditions
  - And virtual distance between client and server
- Server has given throughput
  - Given by HW, our Design, Efficiency, Caching, etc.

CLIENT-SERVER ARCHITECTURE

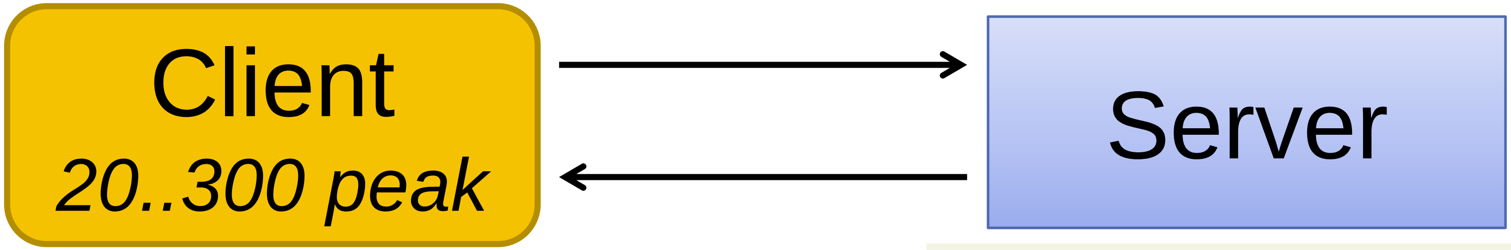


CLIENT-SERVER ARCHITECTURE


Server throughput 300 clients at once

CLIENT-SERVER ARCHITECTURE

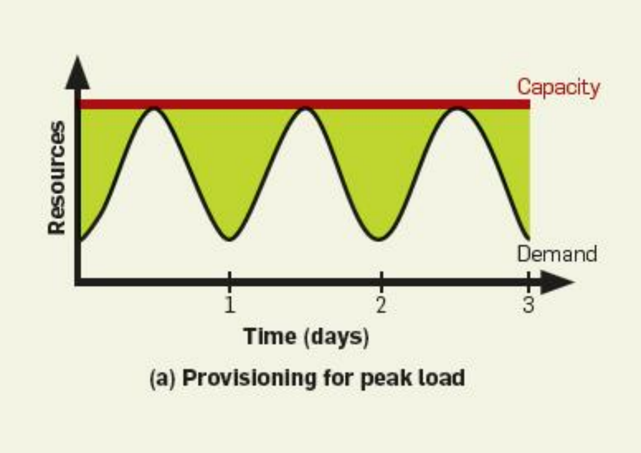
Through put 300
clients at once



Client
20..300 peak

Server

Load grows!



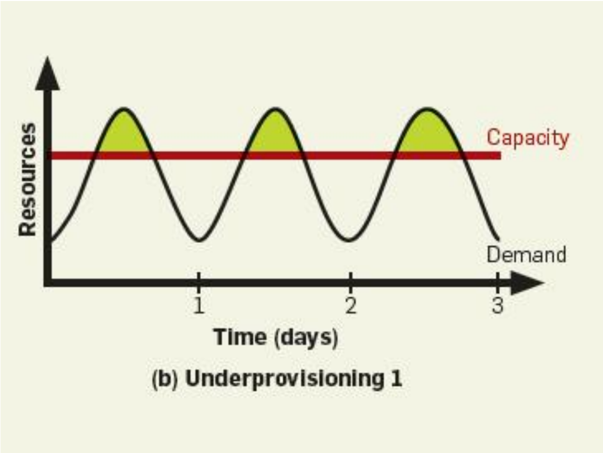
CLIENT-SERVER ARCHITECTURE

Through put 300
clients at once

Client
*20..3**50** peak*

Server

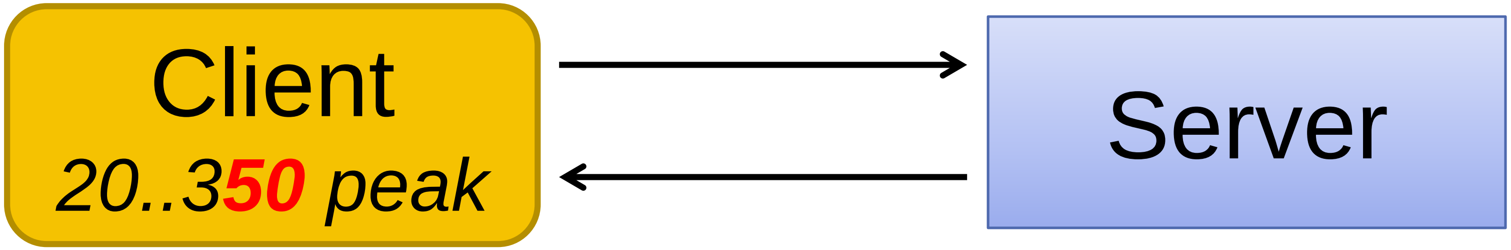
Load grows!



(b) Underprovisioning 1

CLIENT-SERVER ARCHITECTURE

How to improve?



Client

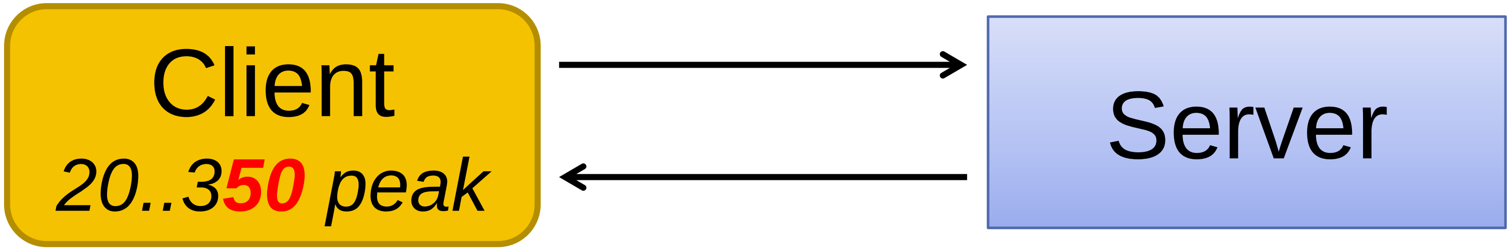
*20..3***50** *peak*

Server

- Caching
- Performance analysis – profiling
- Native/Custom SQL queries for reports
- Better Hardware, more CPU/Mem

CLIENT-SERVER ARCHITECTURE

How to improve?



Client

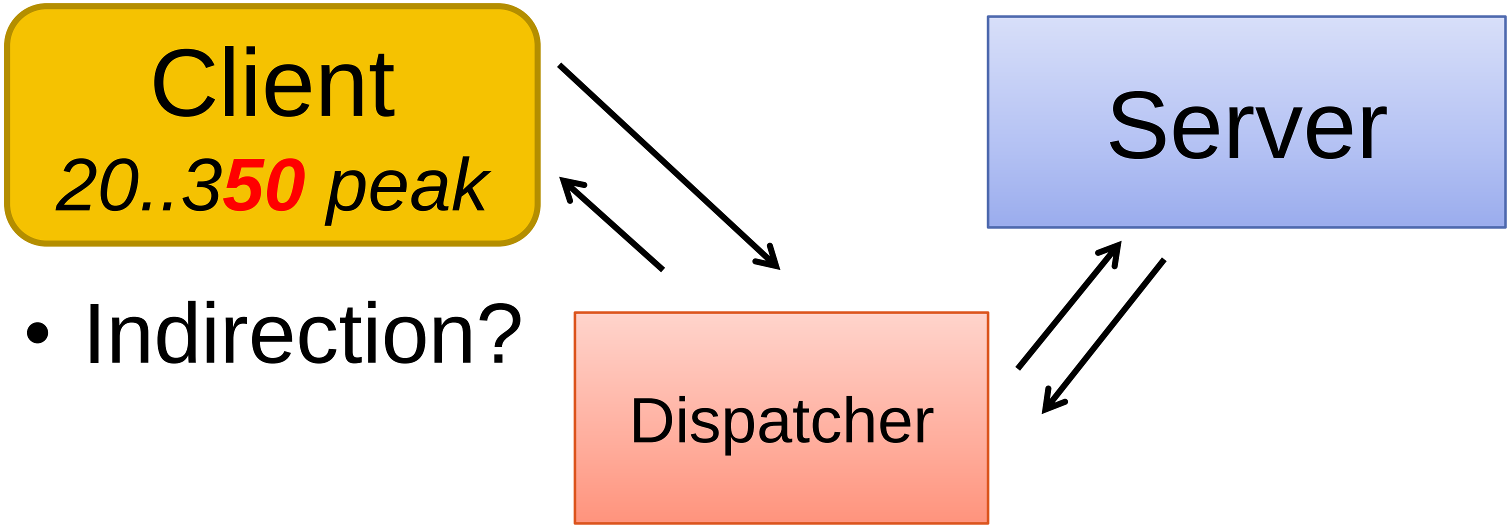
*20..3**50** peak*

Server

- What if it is not enough?
- Indirection?

CLIENT-SERVER ARCHITECTURE

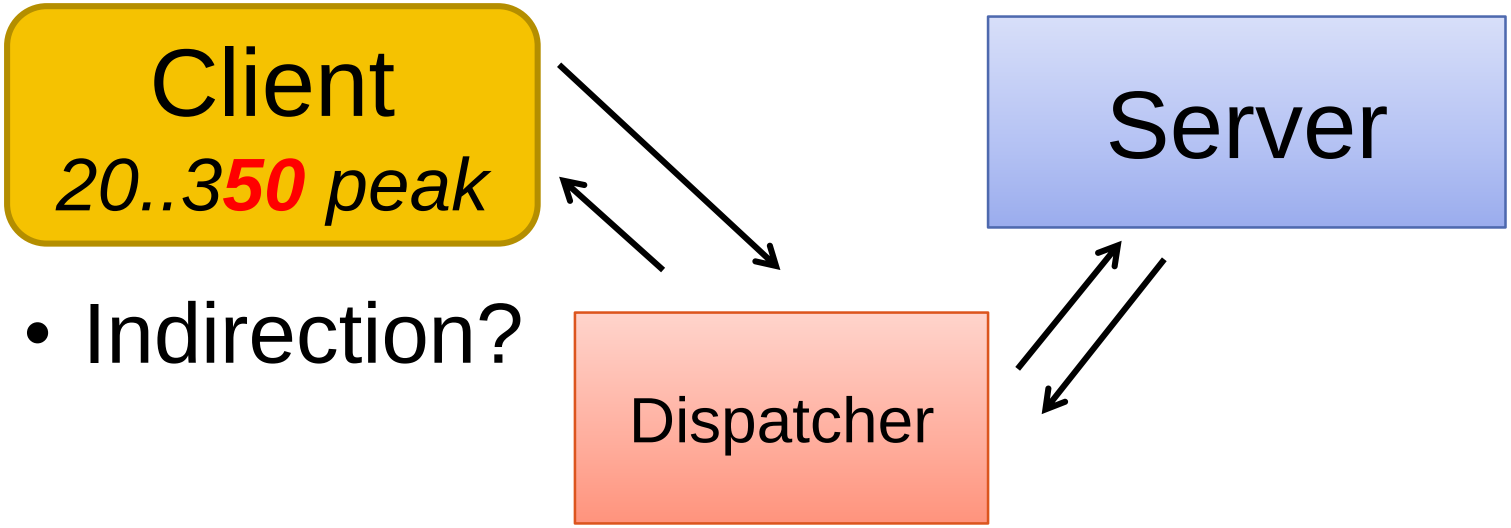
How to improve?



- Indirection?

CLIENT-DISPATCHER-SERVER ARCHITECTURE

How to improve?

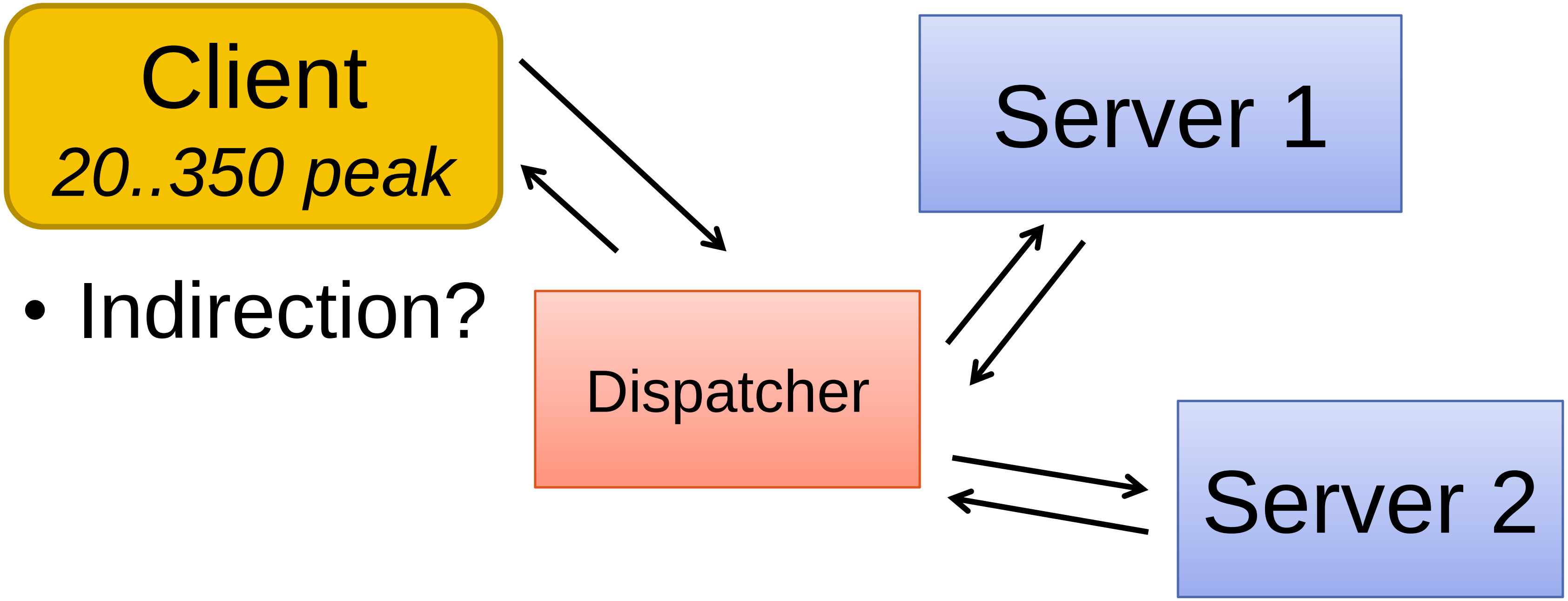


- Indirection?

CLIENT-DISPATCHER-SERVER ARCHITECTURE

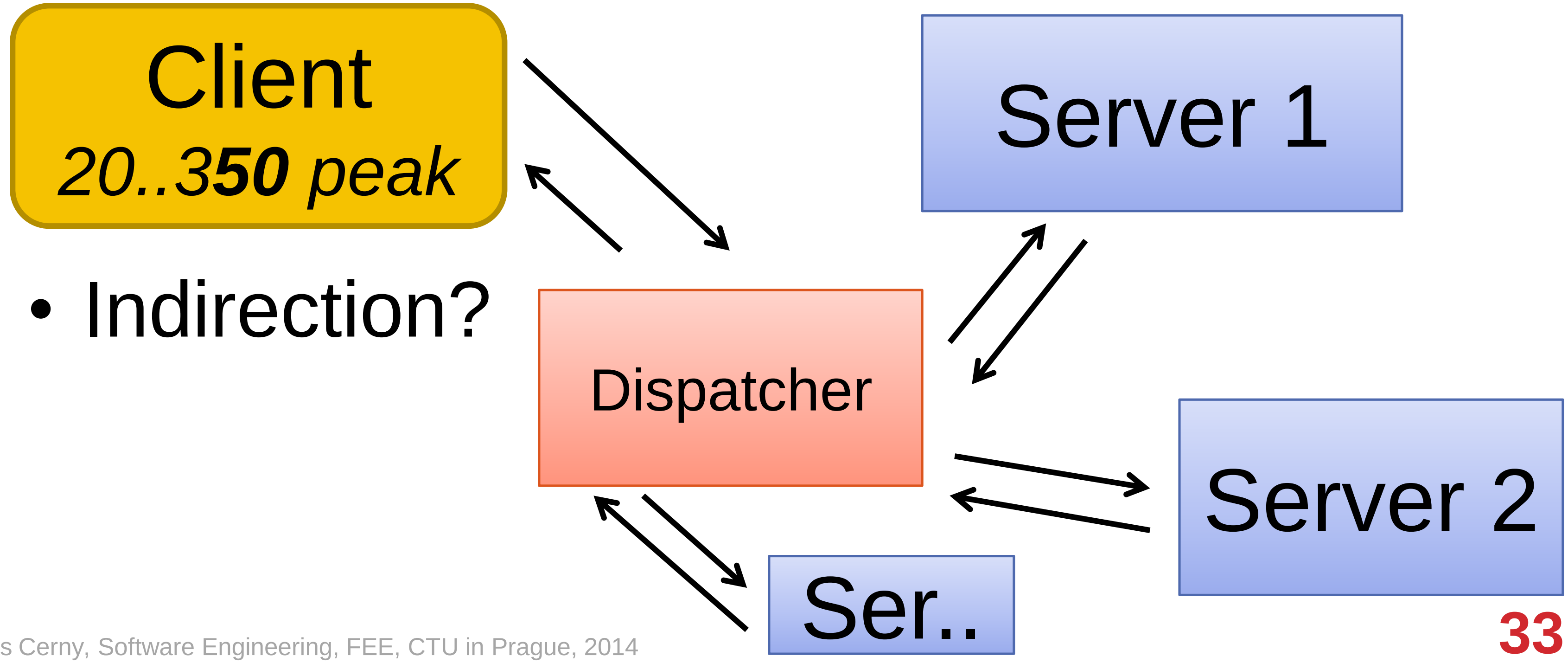
How to improve?

- Indirection?



CLIENT-DISPATCHER-SERVER ARCHITECTURE

How to improve?



Client

*20..**350** peak*

Server 1

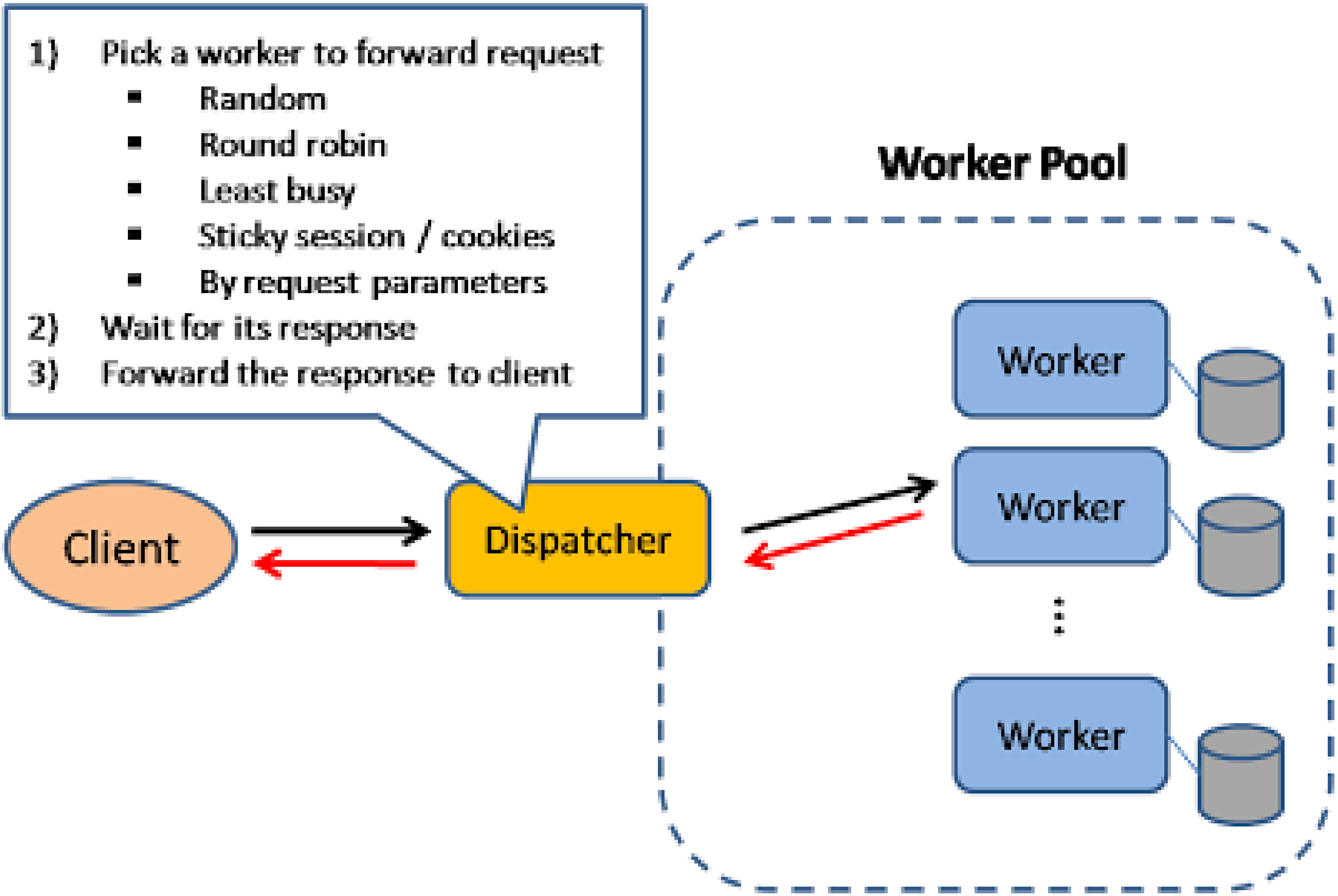
Dispatcher

Server 2

Ser..

- Indirection?

CLIENT-DISPATCHER-SERVER ARCHITECTURE



1) Pick a worker to forward request
   - Random
   - Round robin
   - Least busy
   - Sticky session / cookies
   - By request parameters
2) Wait for its response
3) Forward the response to client

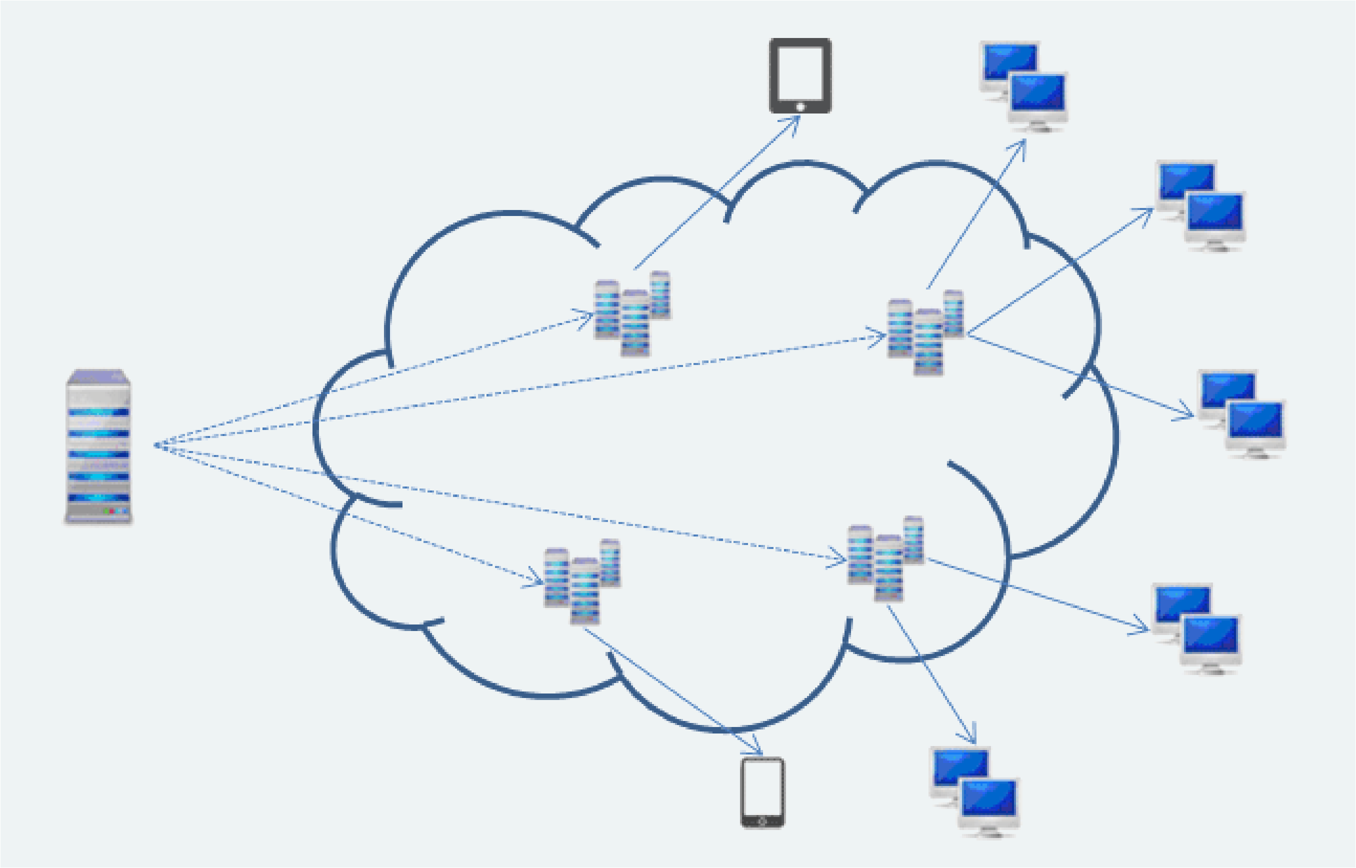
CLIENT-DISPATCHER-SERVER ARCHITECTURE

Most likely we cannot expect to multiply the throughput of the single server

- Balancing overhead
- We can balance different resources
  - Static vs. Dynamic
- Geo-location balancing
  - Content-Delivery-Network (CDN)
    - Static content (Akamai)

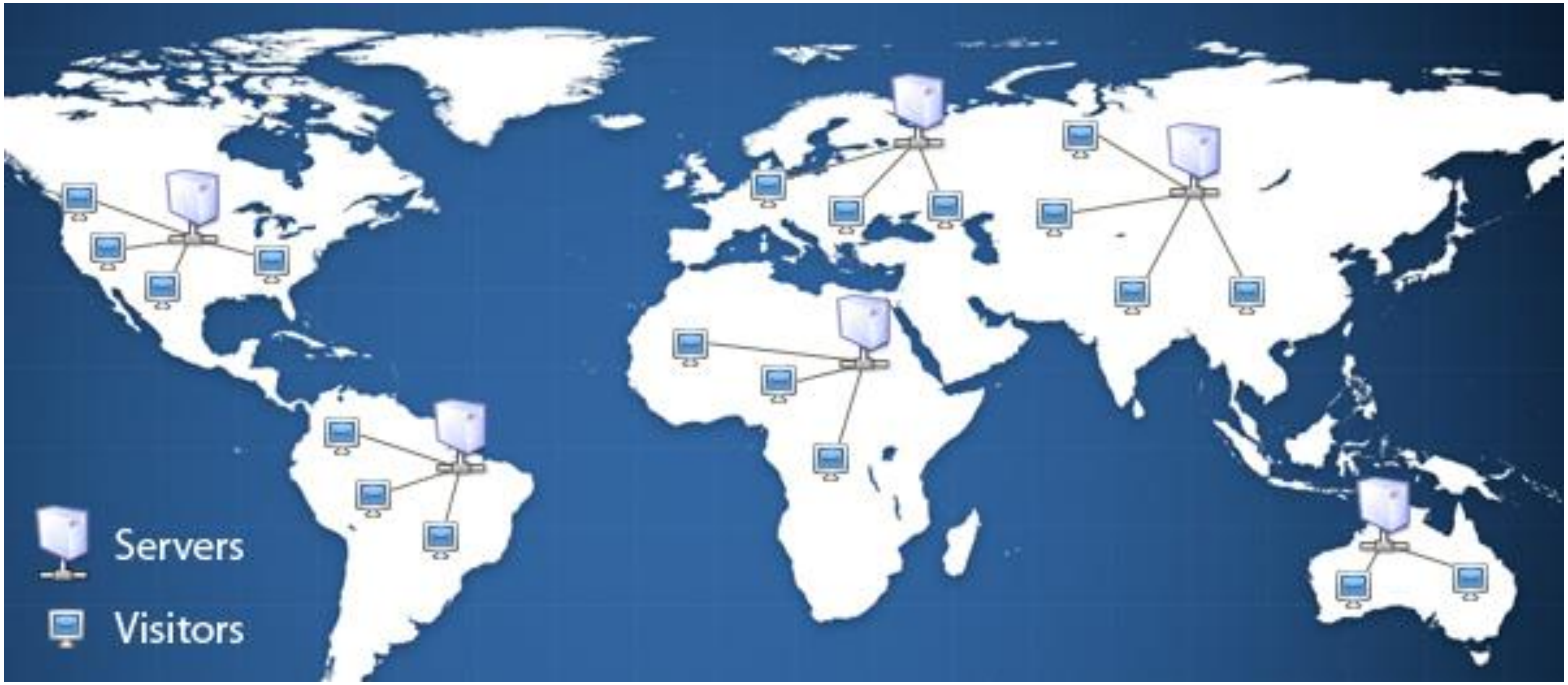
CONTENT DELIVERY NETWORK (CDN)

Example



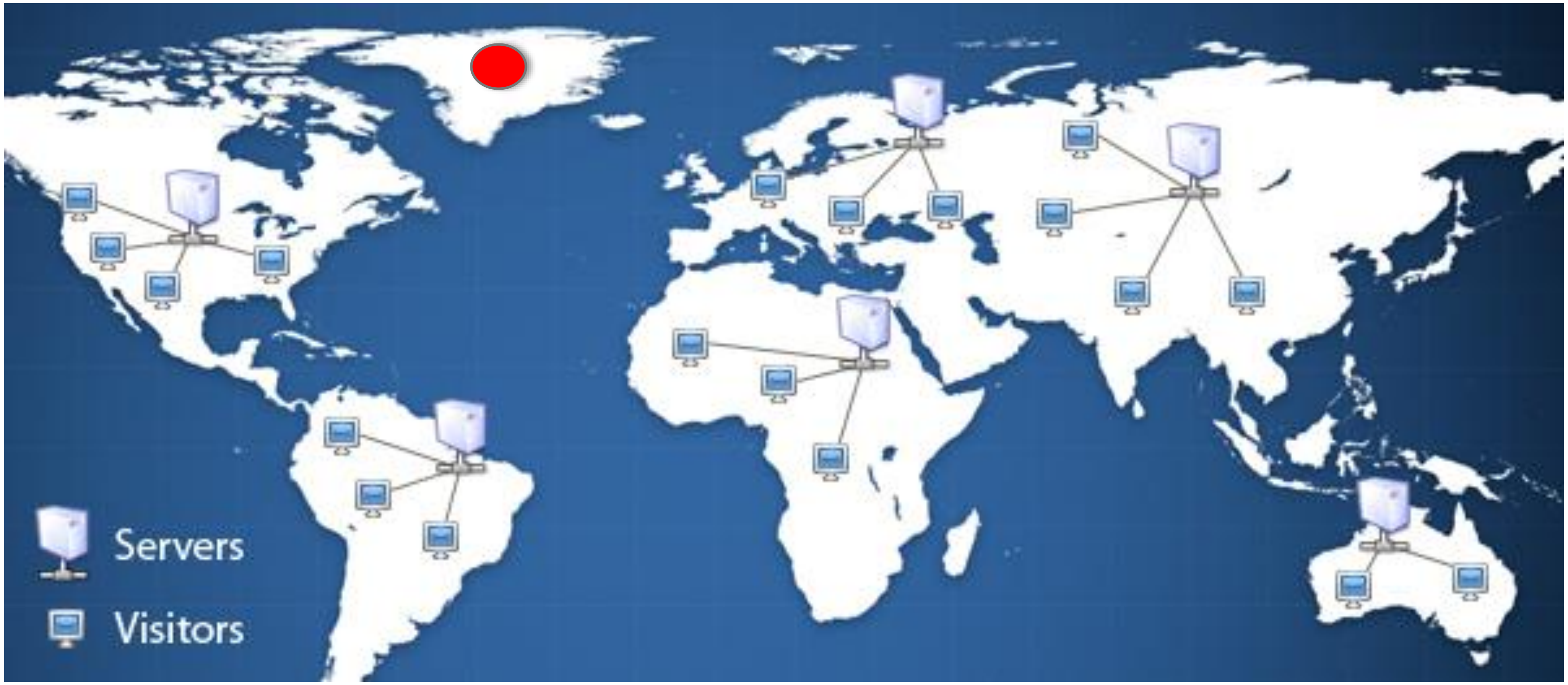
CONTENT DELIVERY NETWORK (CDN)

Example



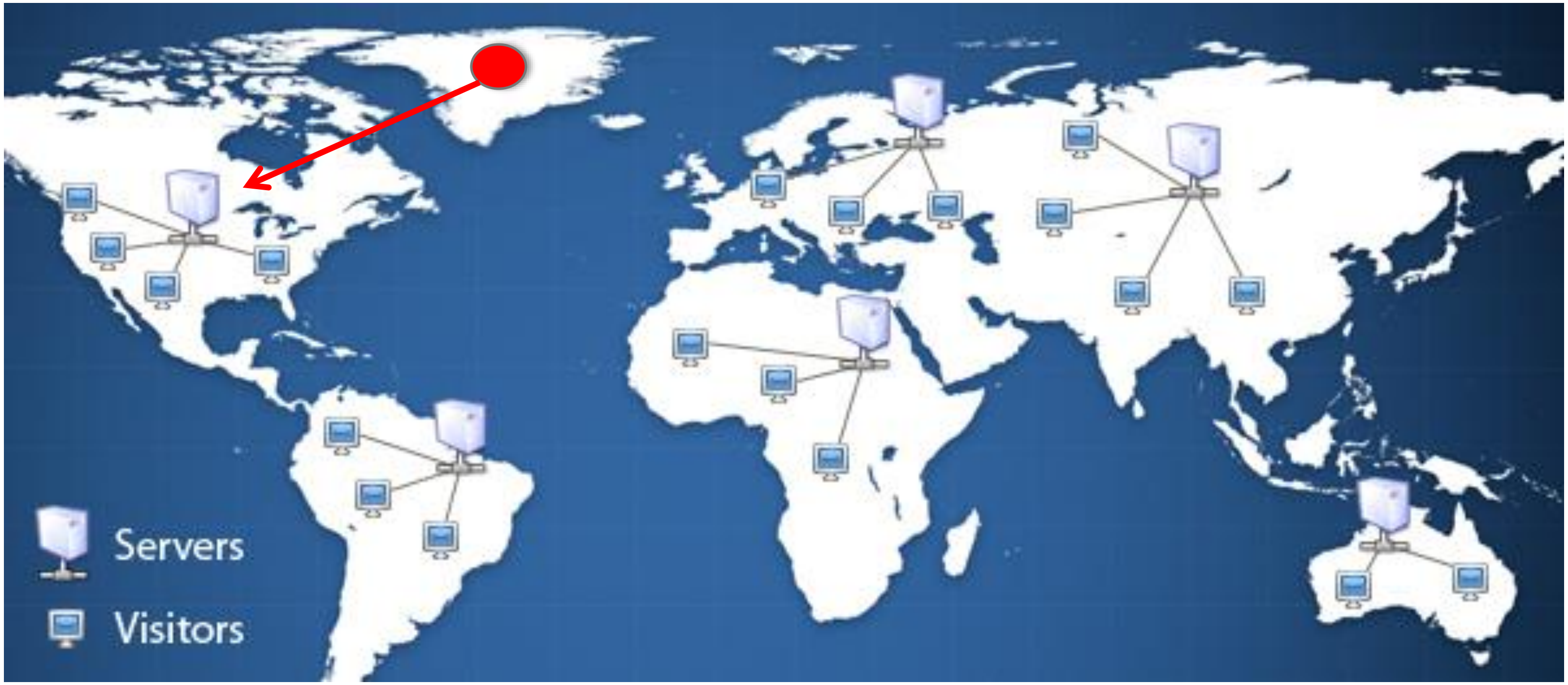
CONTENT DELIVERY NETWORK

Example



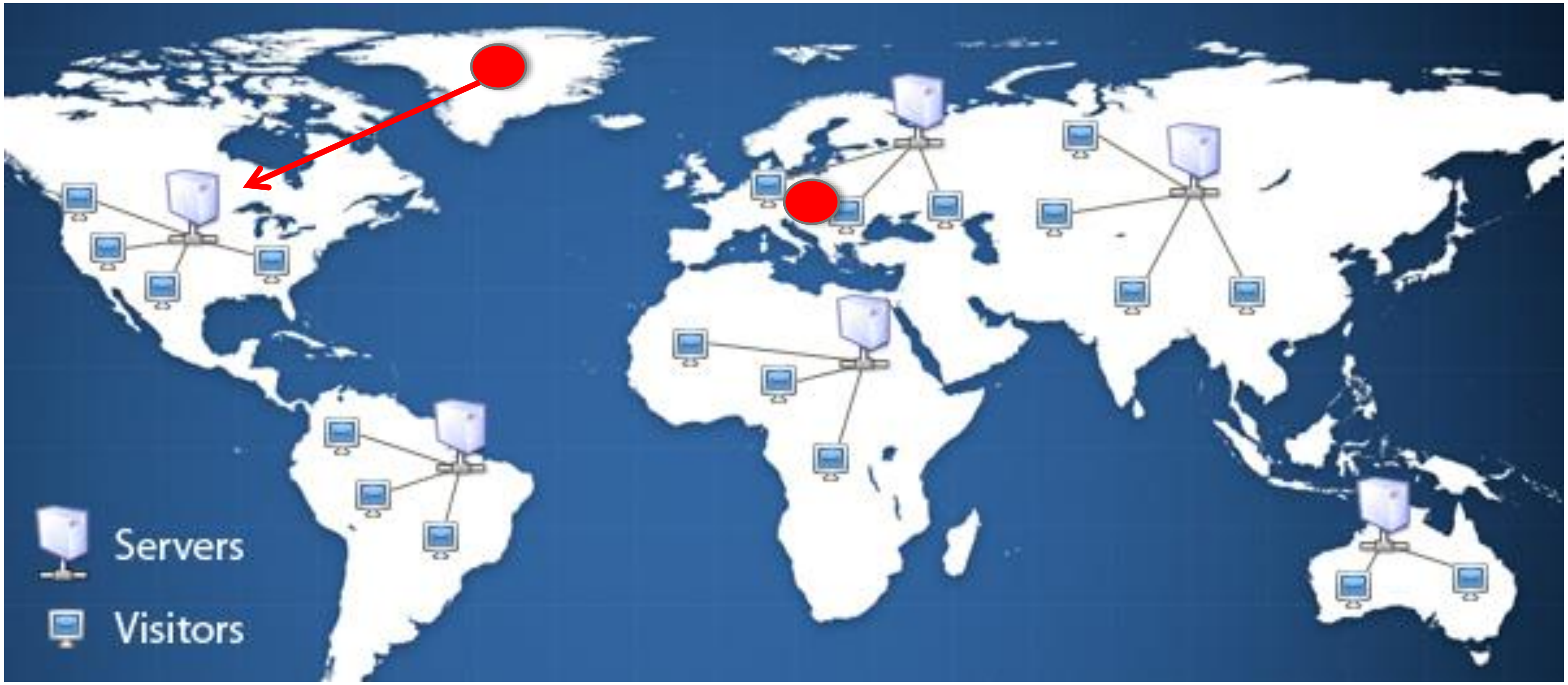
CONTENT DELIVERY NETWORK

Example



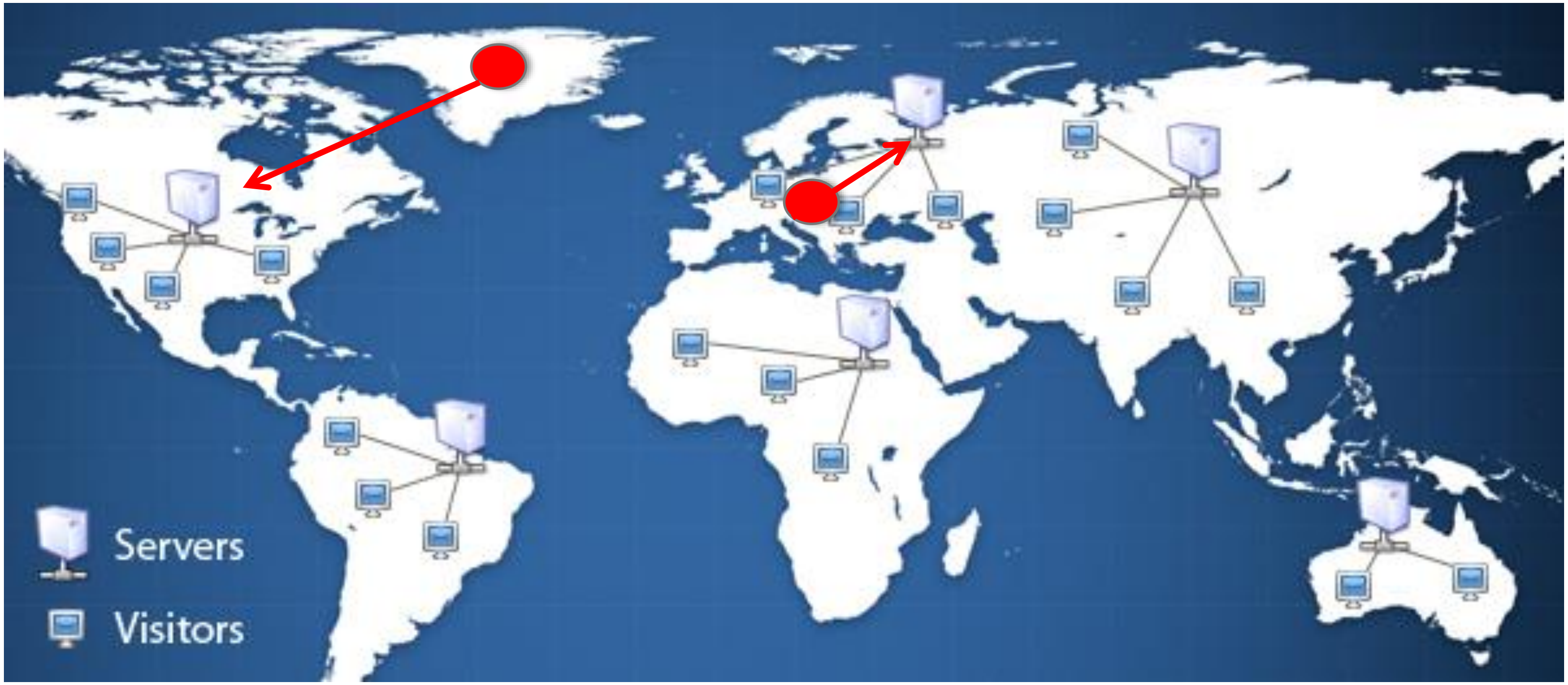
CONTENT DELIVERY NETWORK

Example



CONTENT DELIVERY NETWORK

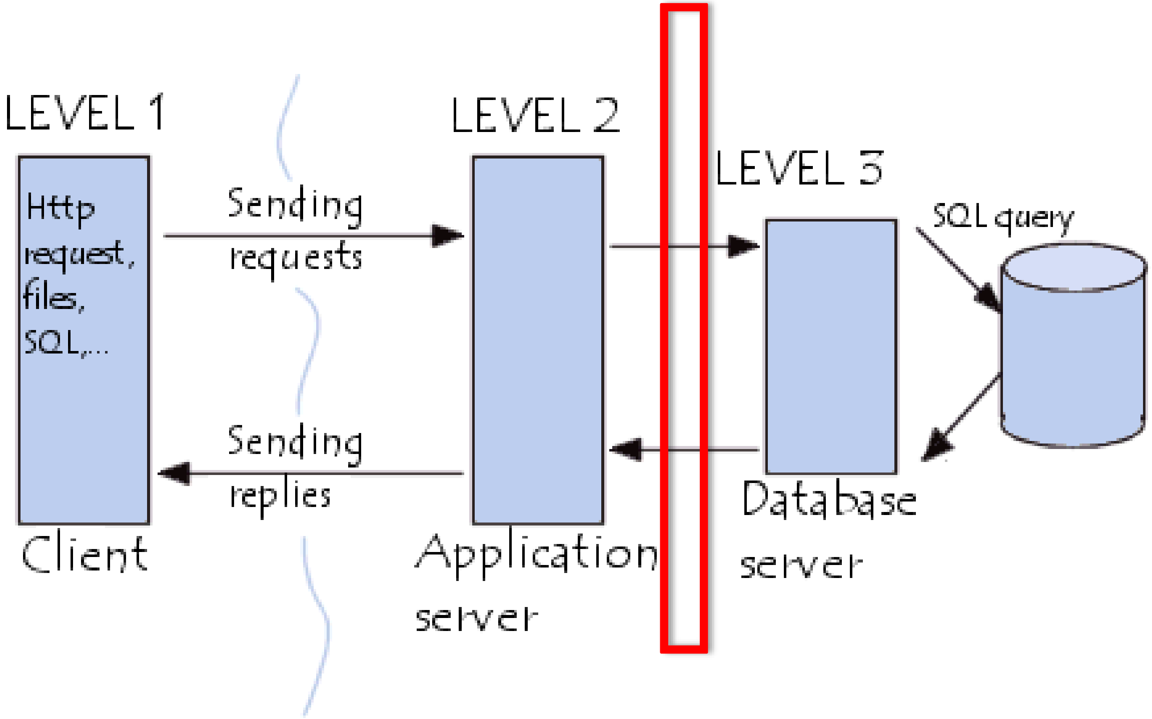
Example



SCALING PERFORMANCE

Database might be the bottleneck

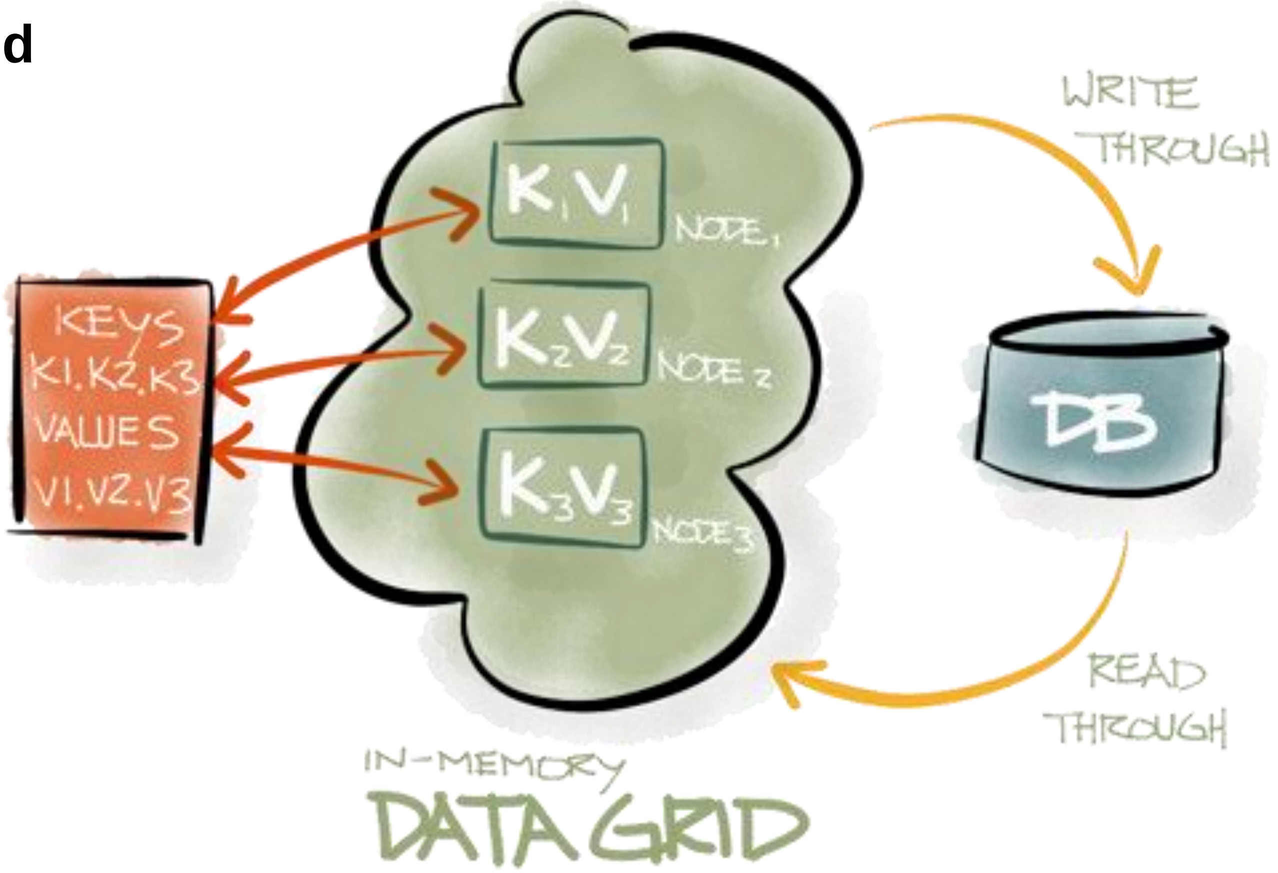
Database replication



SCALING PERFORMANCE

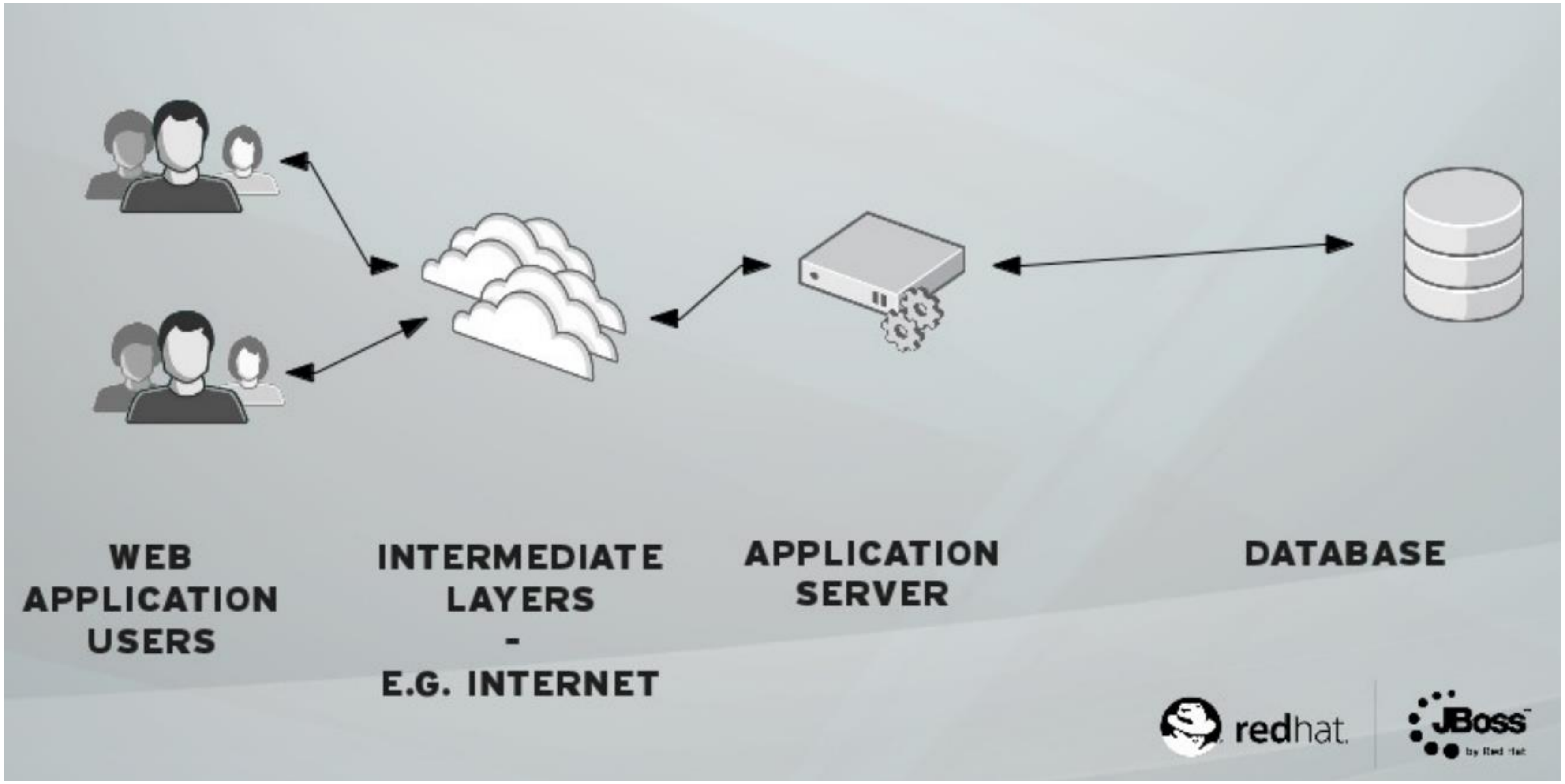
Database might be the bottleneck

Datagrid



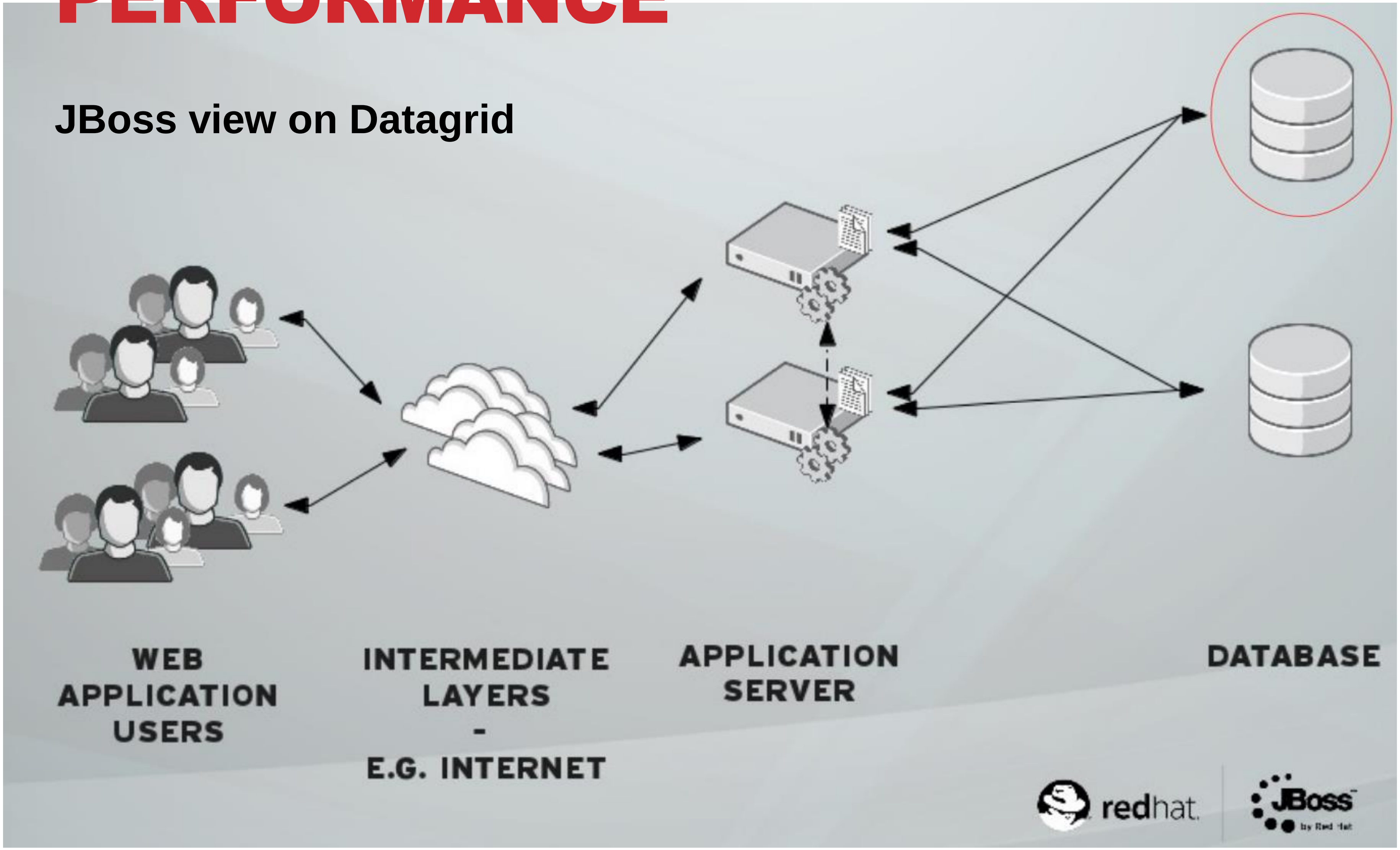
SCALING PERFORMANCE

JBoss view on Datagrid



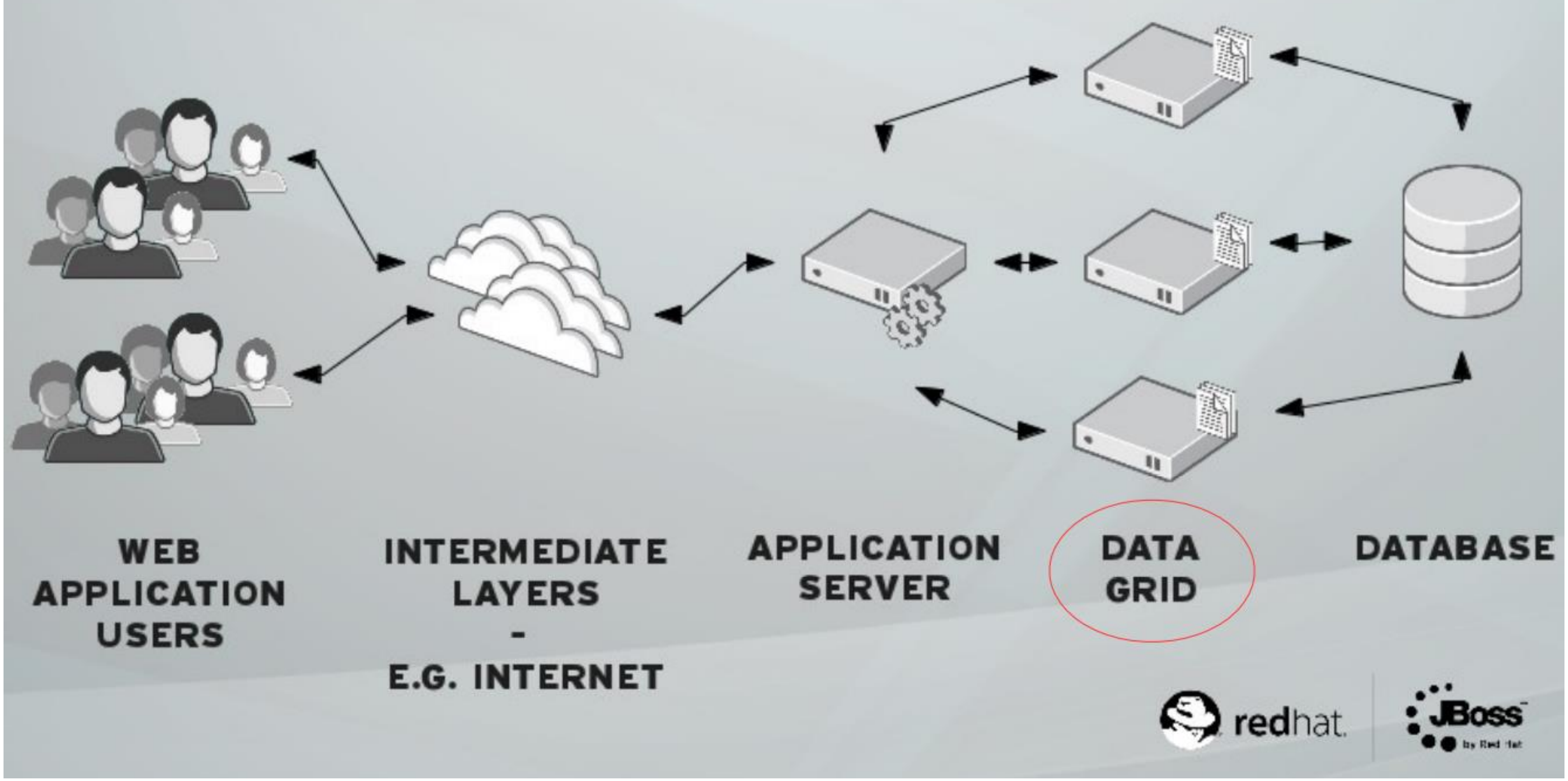
SCALING PERFORMANCE

JBoss view on Datagrid



SCALING PERFORMANCE

JBoss view on Datagrid



SERVICE-ORIENTED ARCHITECTURE (SOA)

So far we considered that server-side app offers data, knowledge and presentation

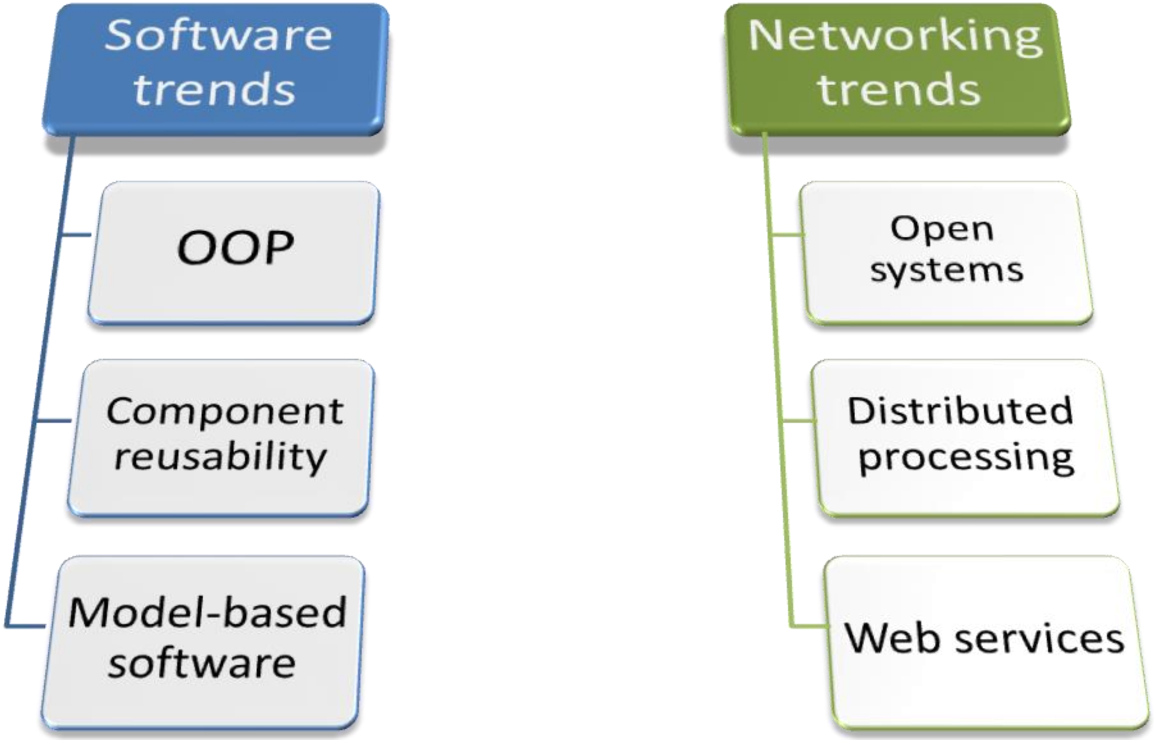
Service does not provide presentation

Well accepted format

- Standard : JSON, SOAP, XML..

SERVICE-ORIENTED ARCHITECTURE

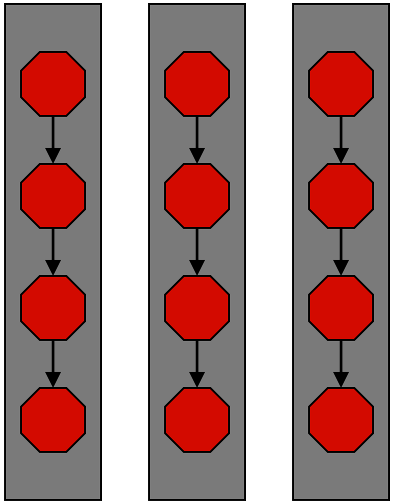
Motivation



- Software trends
  - OOP
  - Component reusability
  - Model-based software
- Networking trends
  - Open systems
  - Distributed processing
  - Web services

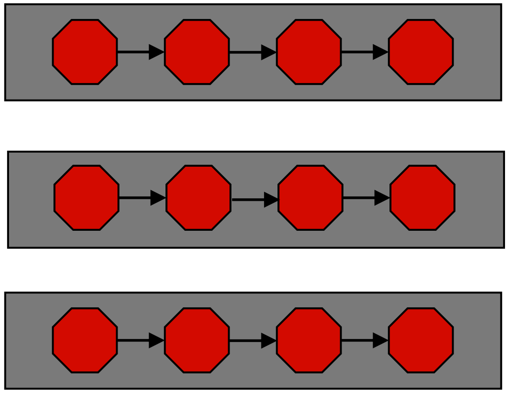
SERVICE-ORIENTED ARCHITECTURE

1960 - 1980



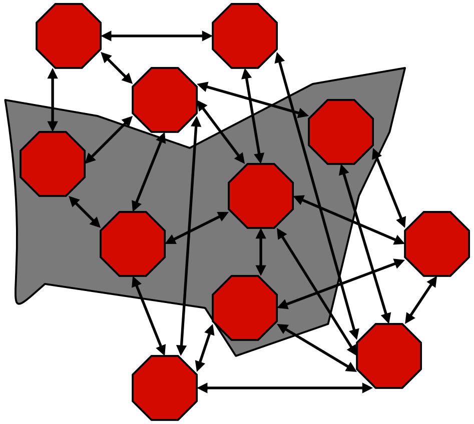
- Organization Focus
- Mainframe Centric
- Internal Use
- Unique Data

1990 - 2000



- Process Focus
- Client Server
- Partial Connectivity
- EDI File Transfer

2010 - 2050

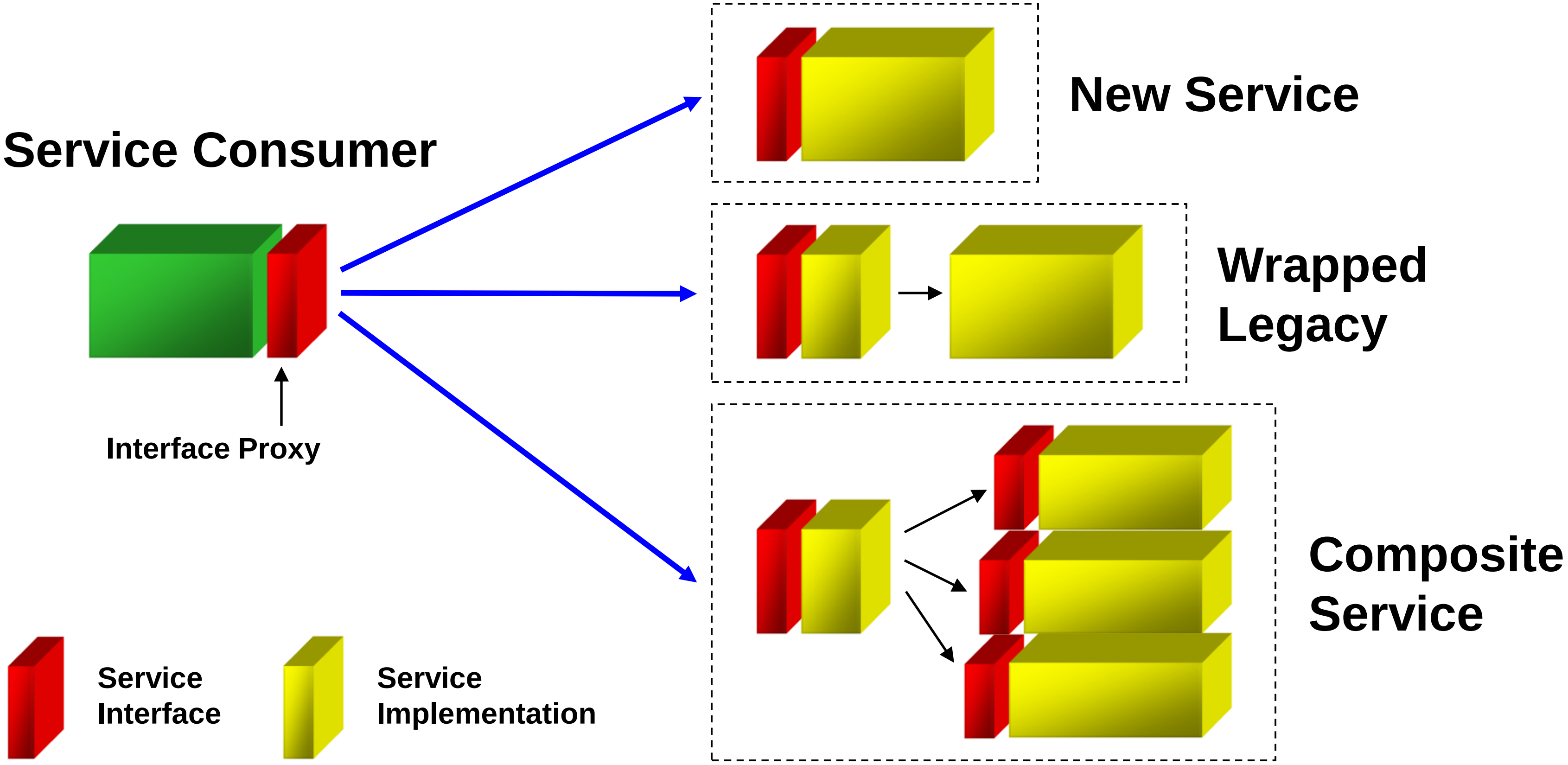


- Distributed Functions
- Data Centric
- Universal Interoperability
- Real-time Connectivity

SERVICE

- Loose coupling
- Reusable
- Stateless
- Autonomous (independent)
- Discoverable
- Abstract
- Composable
- Platform independent

ANATOMY OF A SERVICE



Service Consumer

Interface Proxy

New Service

Wrapped Legacy

Composite Service

Service Interface

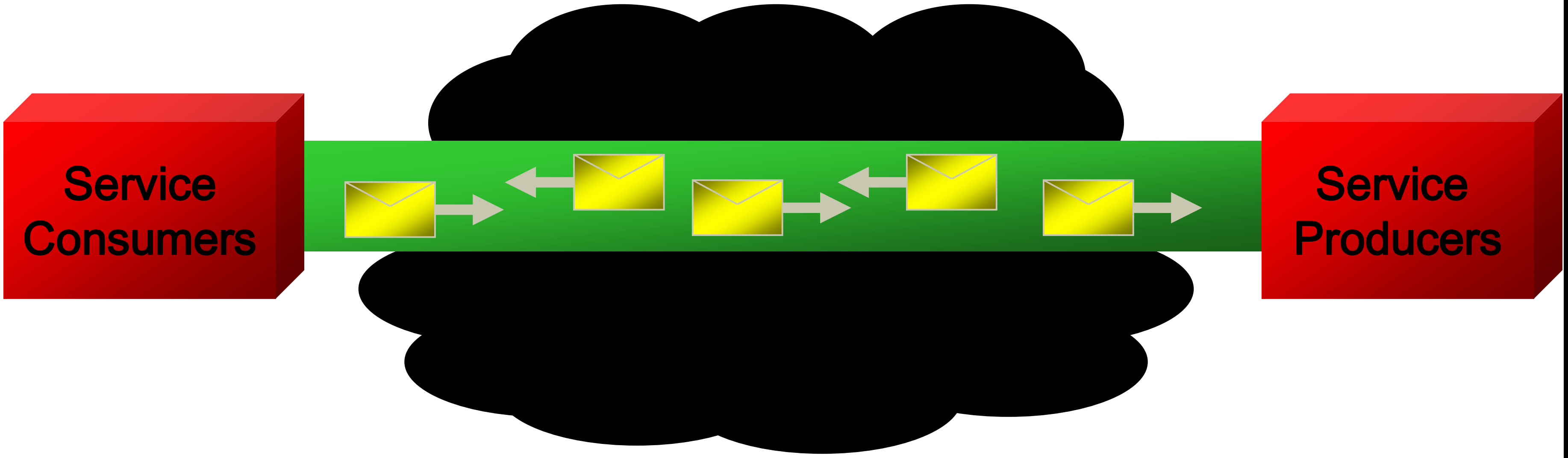
Service Implementation

SERVICES COMMUNICATE WITH MESSAGES

Providing reliability and security to messages

Sending messages across consumers and producers

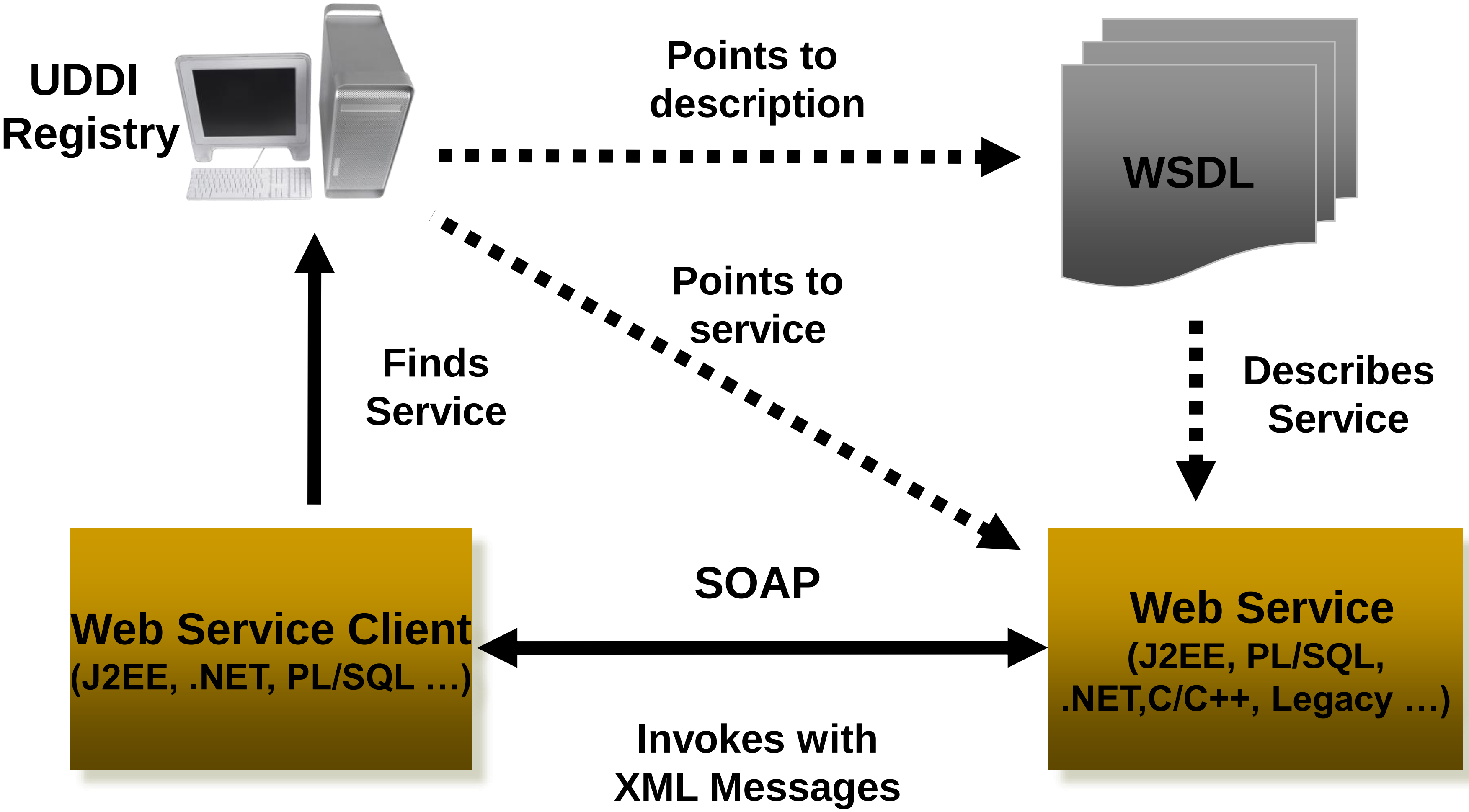
Service Orchestration



Service Consumers

Service Producers

BASIC WEB SERVICES



UDDI Registry

Points to description

WSDL

Finds Service

Points to service

Describes Service

SOAP

Web Service Client
(J2EE, .NET, PL/SQL …)

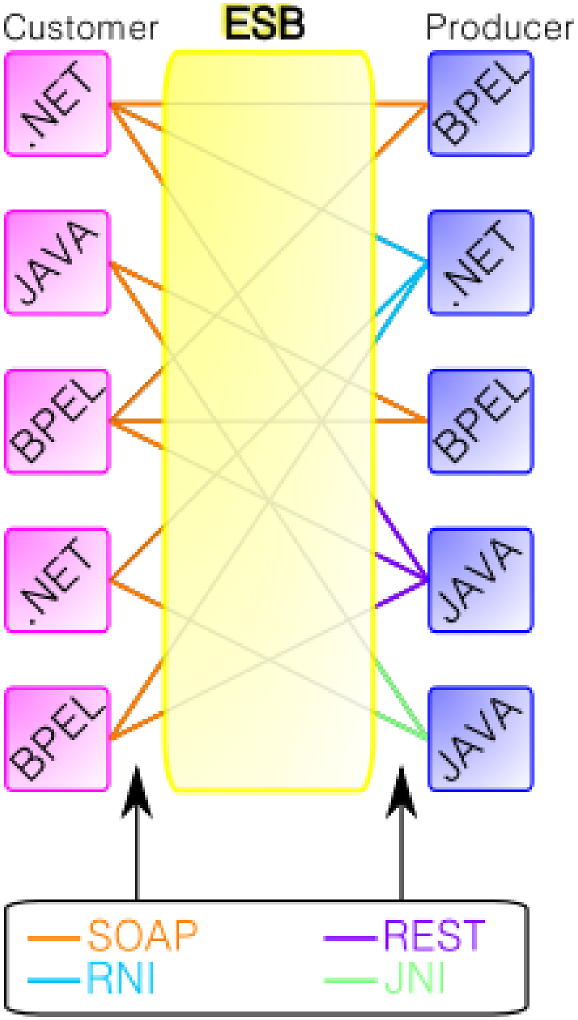
Web Service
(J2EE, PL/SQL, .NET,C/C++, Legacy …)

Invokes with XML Messages

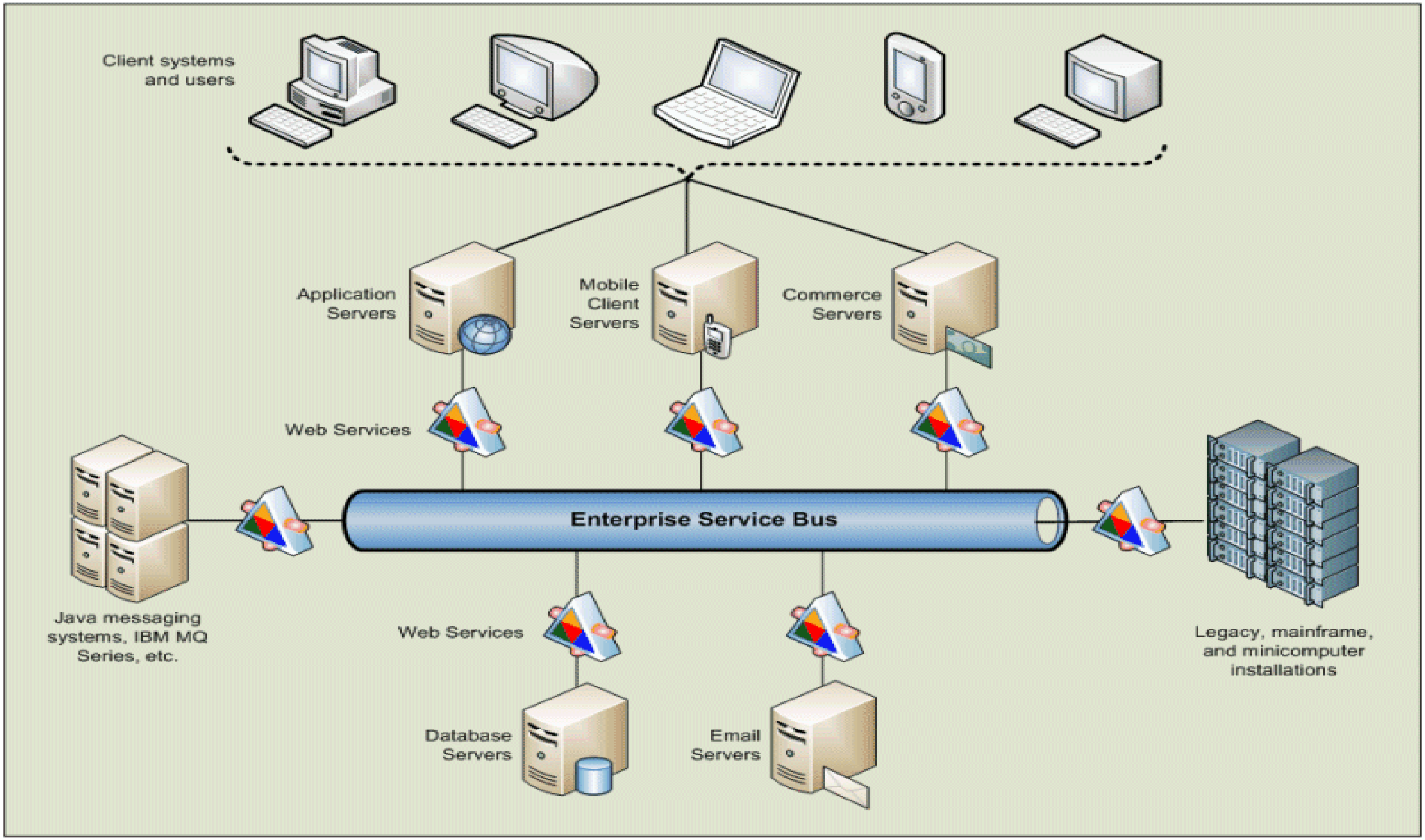
ENTERPRISE SERVICE BUS (ESB)

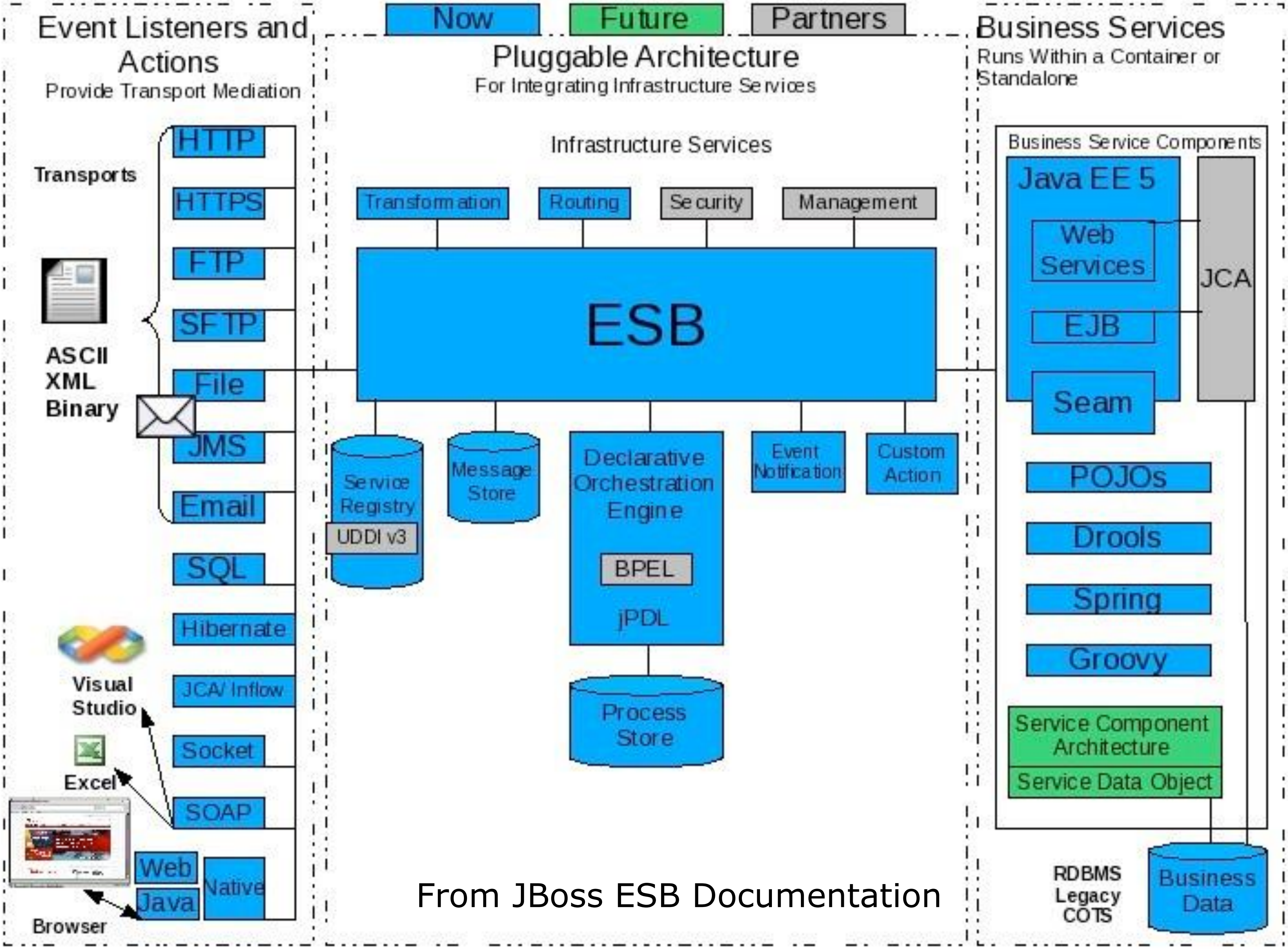
It is **a software architecture model** used for designing and implementing the interaction and **communication** between mutually interacting **software applications in** service-oriented architecture (**SOA**).

- Model for distributed computing
- Variant of client server software architecture model
- Promotes flexibility with regards to communication & interaction between applications.
- Primary use in enterprise application integration (EAI) of heterogeneous and complex landscapes.



ENTERPRISE SERVICE BUS





Now

Future

Partners

Event Listeners and Actions

Provide Transport Mediation

Transports

HTTP

HTTPS

FTP

SFTP

File

JMS

Email

SQL

Hibernate

JCA/ Inflow

Socket

SOAP

Web

Java

Native

ASCII
XML
Binary

Visual Studio

Excel

Browser

Pluggable Architecture

For Integrating Infrastructure Services

Infrastructure Services

Transformation

Routing

Security

Management

ESB

Service Registry

UDDI v3

Message Store

Declarative Orchestration Engine

BPEL

jPDL

Process Store

Event Notification

Custom Action

Business Services

Runs Within a Container or Standalone

Business Service Components

Java EE 5

Web Services

EJB

Seam

JCA

POJOs

Drools

Spring

Groovy

Service Component Architecture

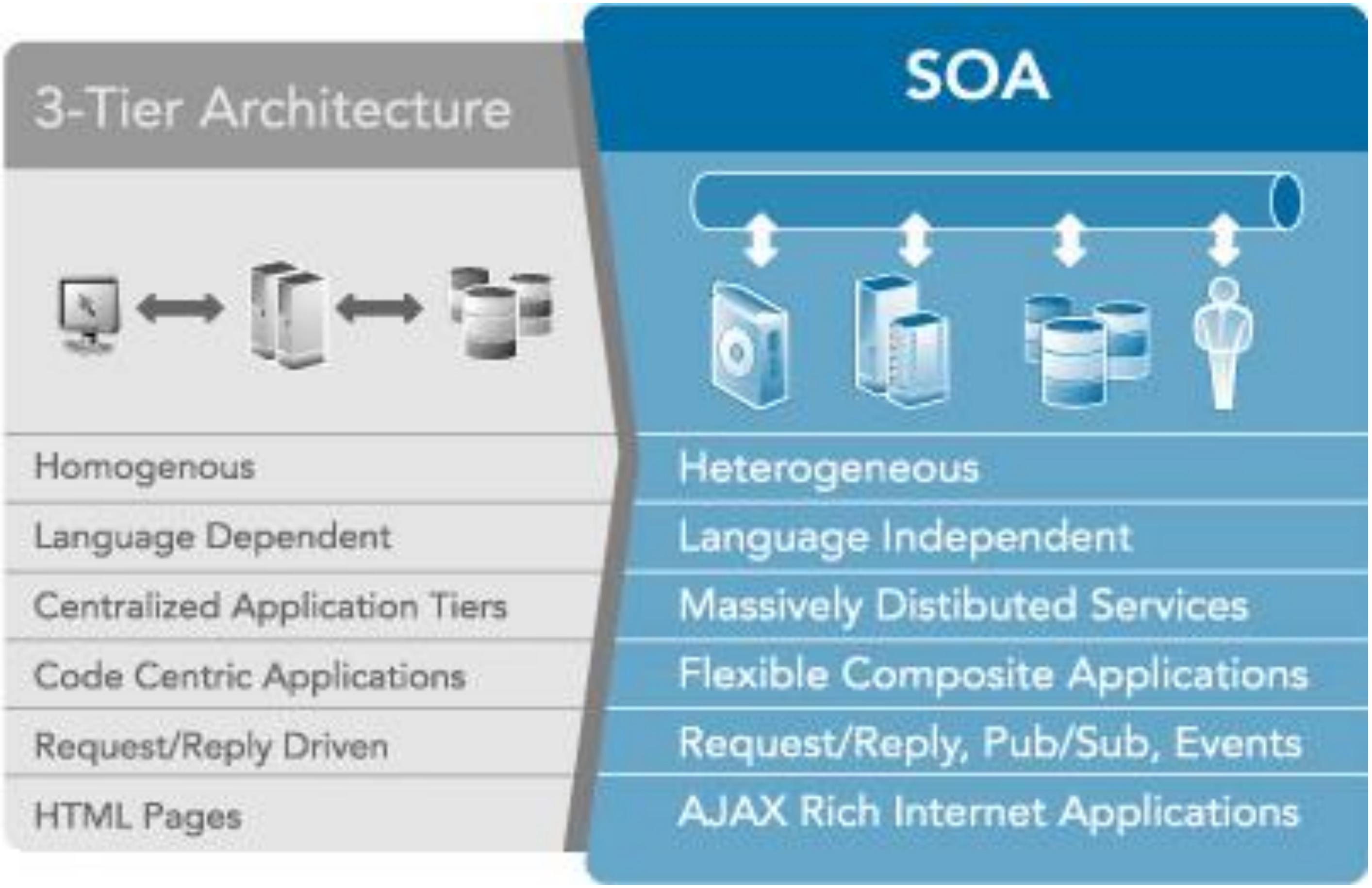
Service Data Object

RDBMS
Legacy
COTS

Business Data

From JBoss ESB Documentation

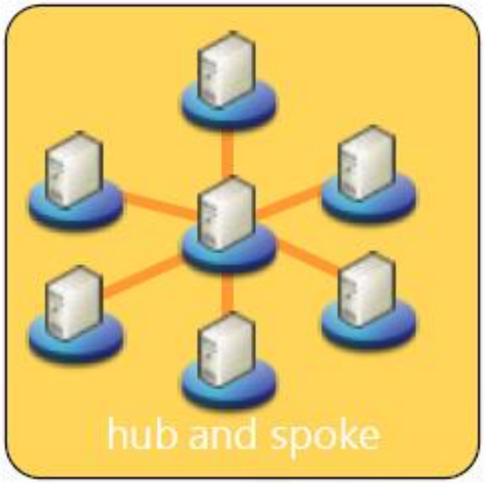
SOA IS AN EVOLUTIONARY STEP



3-Tier Architecture	SOA
Homogenous	Heterogeneous
Language Dependent	Language Independent
Centralized Application Tiers	Massively Distibuted Services
Code Centric Applications	Flexible Composite Applications
Request/Reply Driven	Request/Reply, Pub/Sub, Events
HTML Pages	AJAX Rich Internet Applications

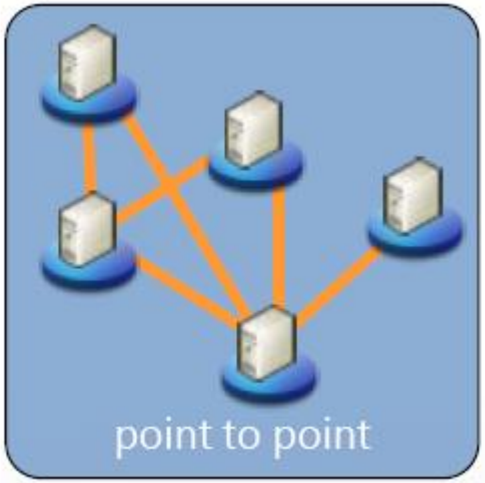
SOA IS AN EVOLUTIONARY STEP

in distributed communications



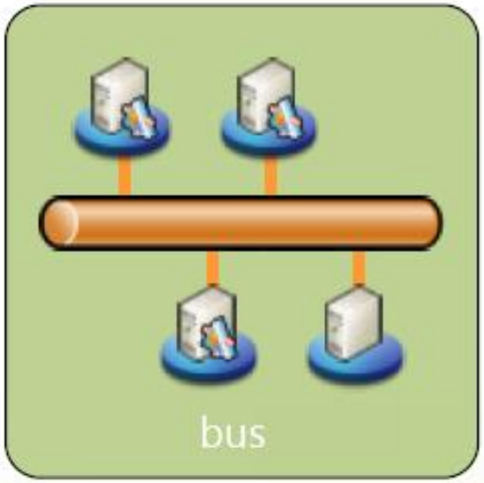
“too centralized

EAI



“too decentralized”

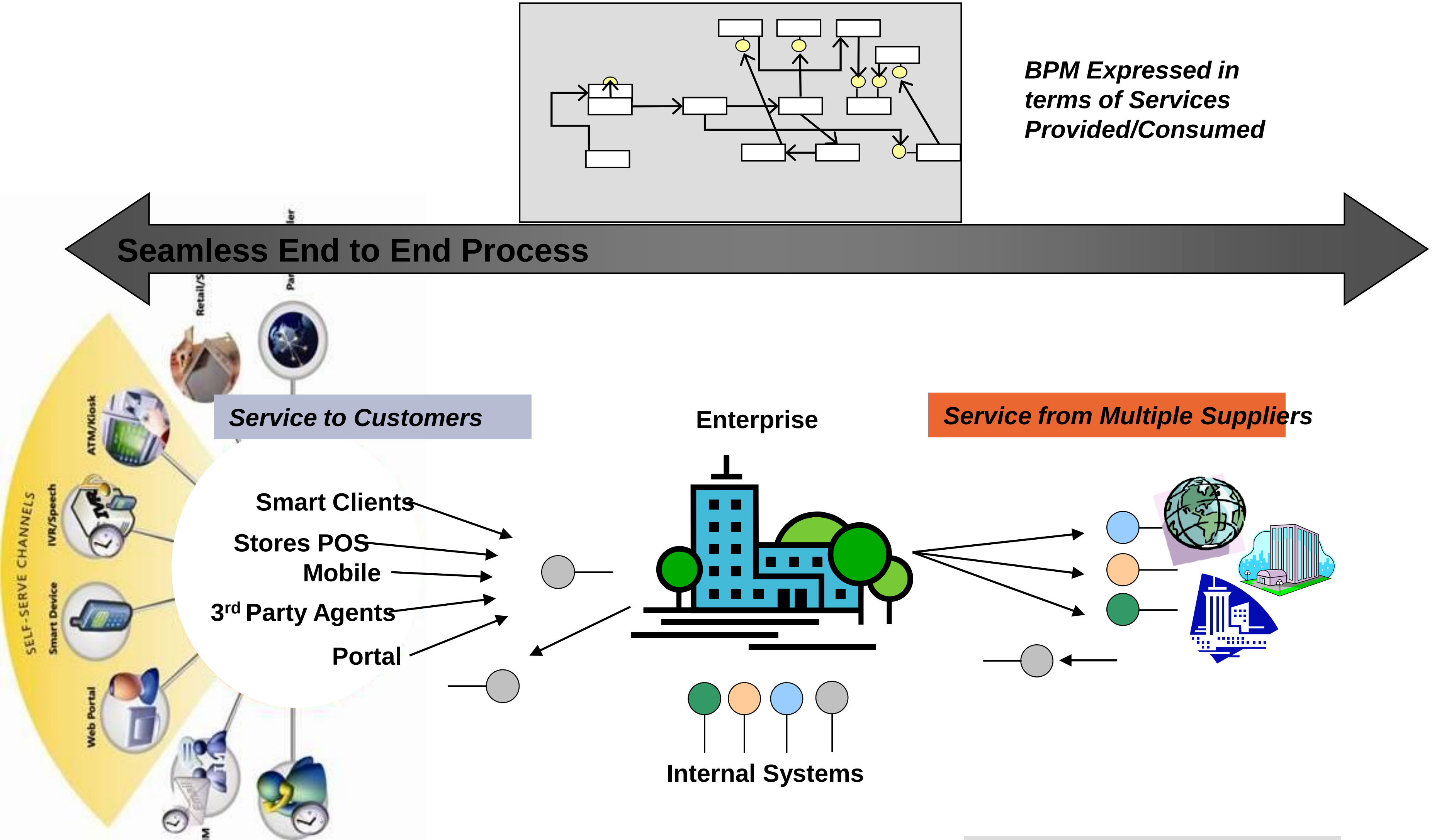
Project-ware



“just right”

SOA

TO ENABLE BUSINESS PROCESS OPTIMIZATION AND THE REAL TIME ENTERPRISE (RTE)



BPM Expressed in terms of Services Provided/Consumed

Seamless End to End Process

Service to Customers

Smart Clients

Stores POS

Mobile

3rd Party Agents

Portal

Enterprise

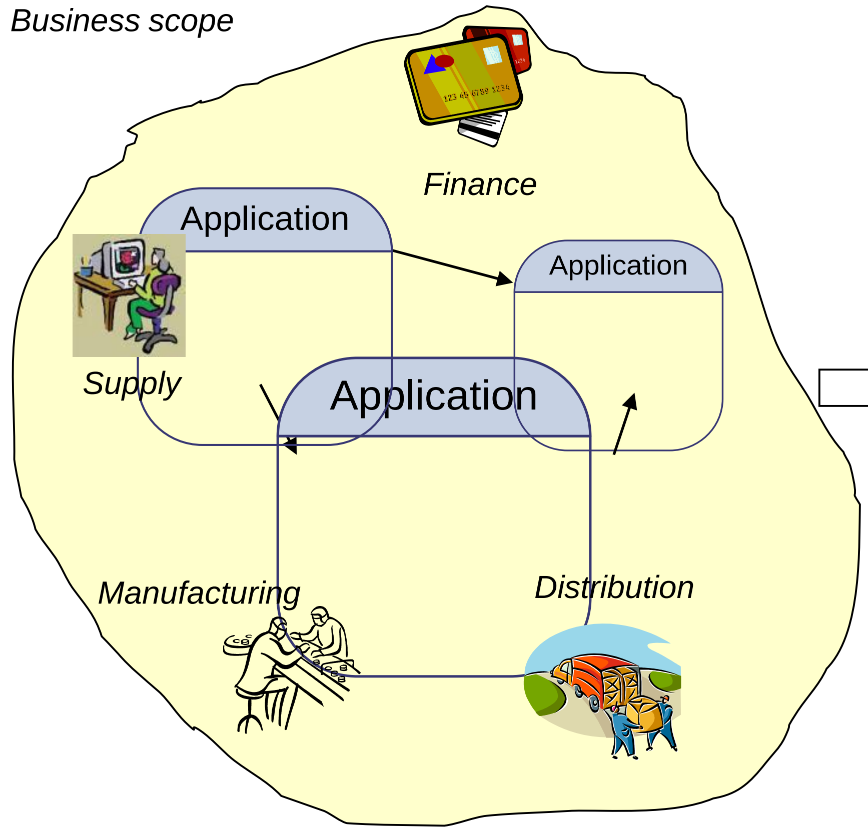
Internal Systems

Service from Multiple Suppliers

SOA Patterns: Single, Multi-Channel Service for consistency

SOA Pattern: Standardized Service provided by multiple suppliers

APPLICATION CENTRIC



Business scope

Finance

Application

Application

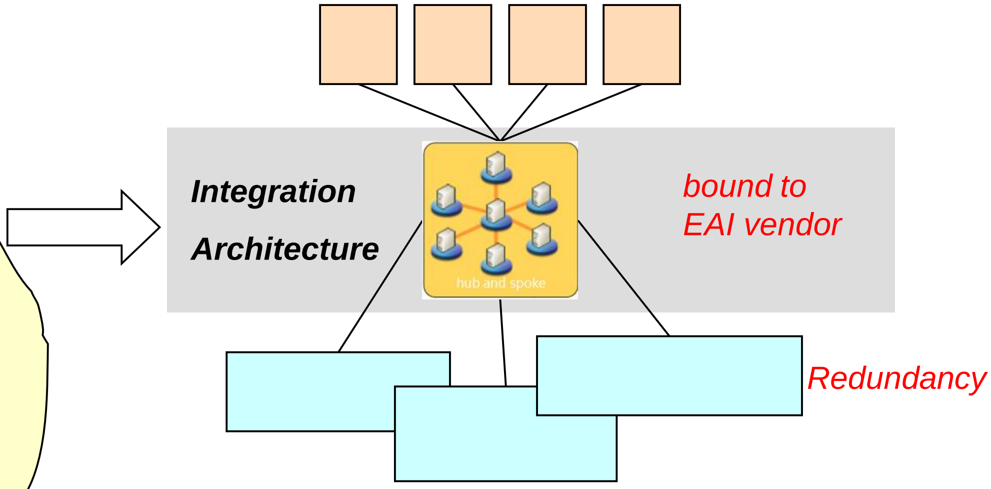
Supply

Application

Manufacturing

Distribution

Business functionality is duplicated in each application that requires it.



Narrow Consumers
Limited Business Processes

Integration

Architecture

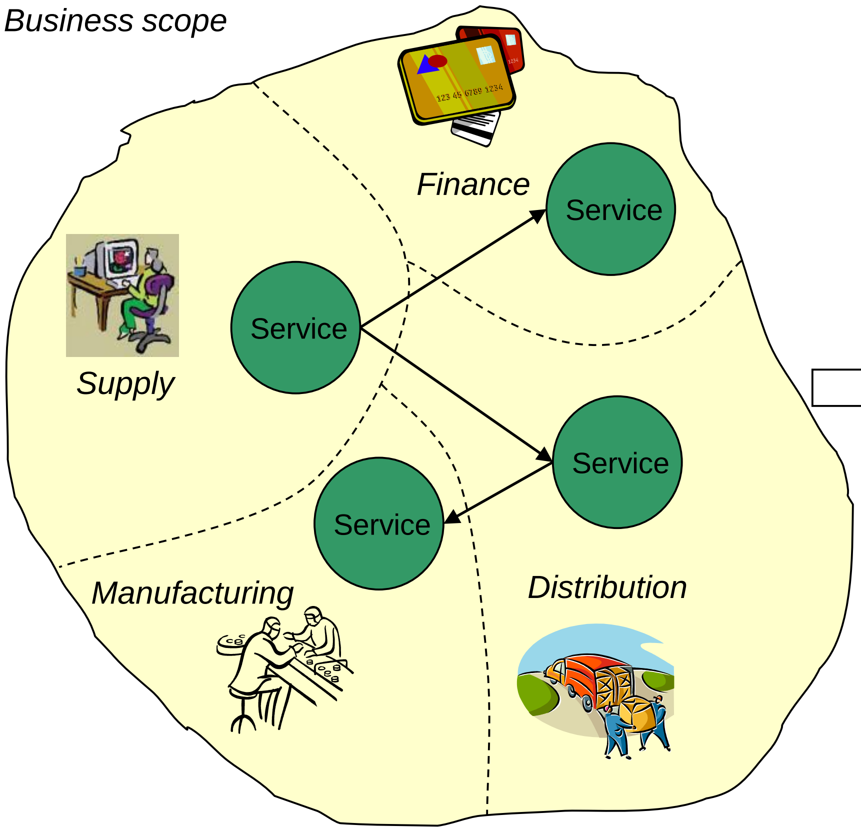
bound to
EAI vendor

Redundancy

Overlapped resources
Overlapped providers

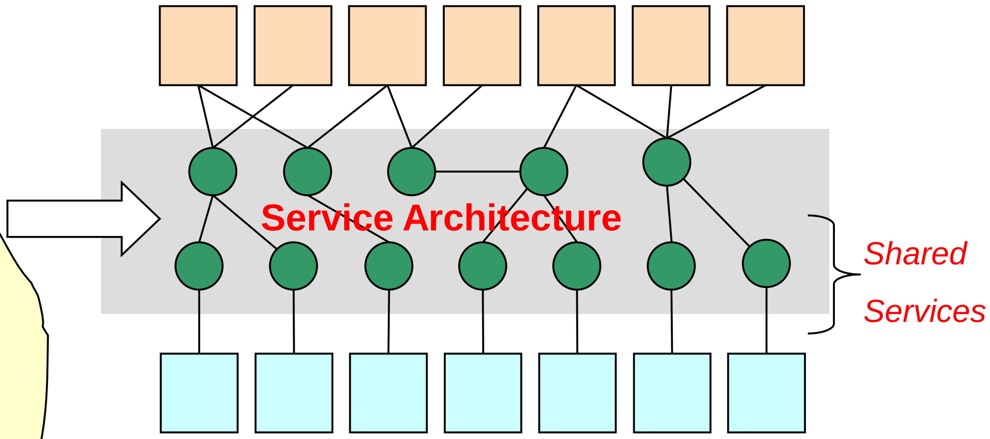
EAI ‘leverage’ application silos with the drawback of data and function redundancy.

SERVICE CENTRIC



SOA structures the business and its systems as a set of capabilities that are offered as Services, organized into a Service Architecture

Multiple Service Consumers
Multiple Business Processes



Multiple Discrete Resources
Multiple Service Providers

Service virtualizes how that capability is performed, and where and by whom the resources are provided, enabling multiple providers and consumers to participate together in shared business activities.

SERVICE CENTRIC APPROACHES

Open your business to extension and evolution!

Natural extension and reuse

- Expedia API, Paypal, Amazon API, Airfare, Heureka..

Open your system to novel needs, requirements, interaction

Reuse by other vendors!