Middleware

TECHNISCHE UNIVERSITÄT DARMSTADT

A. Buchmann
Wintersemester 2014/2015

Course Organization

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lecture:
  - Prof. A. Buchmann: buchmann@dvs.tu-darmstadt.de, Phone: 6230
    - By appointment, preferably Monday 16:30 to 18:00
    - Appointments with Mrs. Tiedemann – tiedem@dvs.tu-darmstadt.de
- Exercises:
  - Sebastian Frischbier: frischbier@dvs.tu-darmstadt.de, Phone: 6231
  - Tobias Freudenreich: freudenreich@dvs.tu-darmstadt.de, Phone: 3413

- Course web page: http://www.dvs.tu-darmstadt.de/teaching/cs/
  - Material updated on regular basis. Please check frequently.
  - User: mw11 Password: middleware@3413

- Time and Place:
  - Lecture: Wed. 16:15-17:55 in S202/C110
  - Exercise: Tue. 17:05-18:45 in S202/C120

Goals of the course

TECHNISCHE UNIVERSITÄT DARMSTADT

- This is a grand tour of architectures for large distributed systems
- Major concepts and problems in distributed systems
  - Principles
  - Solution techniques
  - Middleware
  - Technological overview
- Exercises: papers and discussions, summary and answers to questions about papers graded pass/fail, practical project
- You should gain an understanding of how large real-life systems work.
- **Required: Interest and commitment to learn, read and experiment!**

Grading

TECHNISCHE UNIVERSITÄT DARMSTADT

- Exam may cover material from lectures and exercises
- Bonus of up to 1.0 grades is possible based on exercises and project
- Exercises incl. project (maximum 100 pts.):
  - Papers and discussions 40% (4 papers, answers graded on pass/fail, some questions in exam)
  - Project 60% based on milestones reached
- Thresholds for bonus:
  - 76-85 pts. 1/3 grade
  - 86-95 pts. 2/3 grade
  - 96-100 pts. 3/3 grade

Middleware Exercises: Names & Faces

TECHNISCHE UNIVERSITÄT DARMSTADT

Tobias Freudenreich

Contact: *lastname*@dvs.tu-darmstadt.de

Middleware Exercises: Rules of the Game

TECHNISCHE UNIVERSITÄT DARMSTADT

- We request submissions with answers to the questions about the assigned papers
  - …we will grade them on a pass/fail basis
  - …we ask you to participate actively in the discussions
  - …you are encouraged to present and discuss your solutions to the various aspects of the project to the group

- You can work in teams but remember that you take the exam alone
  - Teams should have at least 2 and at most 3 members
  - Work is calculated so that a 2-person team can do a good job
  - You will form the teams on your own. The teams are fixed with submission of the first deliverable / milestone
  - Team members will be selected at random to present the team's solution

Acknowledgements

The course has been evolving over the past 10 years

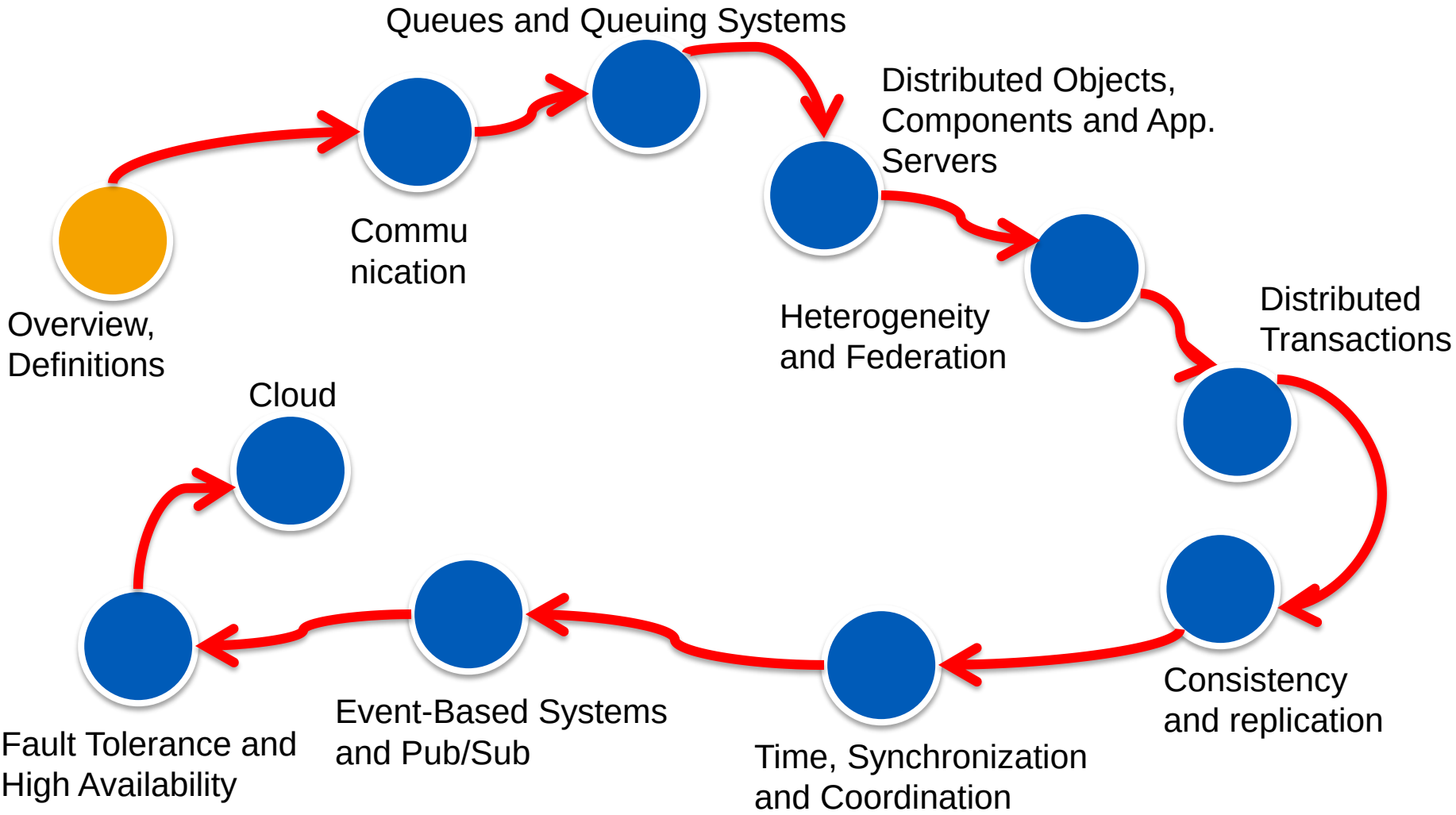
Due to the fast paced development of the area, every year a substantial portion of the course must be reworked

Special thanks to Prof. Dr. Ilia Petrov (Reutlingen) who produced the set of slides for the course offered WS 2011/2012. These slides are the basis for this year's course

I also wish to acknowledge Dr. Mariano Cilia (Intel Corp. Argentina) and Prof. Gustavo Alonso (ETH Zürich) who provided material for earlier versions that has been reused here

Acknowledgement of the source will be provided where possible/practical

Topics


Queues and Queuing Systems
Distributed Objects, Components and App. Servers
Commu nication
Overview, Definitions
Heterogeneity and Federation
Distributed Transactions
Cloud
Consistency and replication
Fault Tolerance and High Availability
Event-Based Systems and Pub/Sub
Time, Synchronization and Coordination

Course Materials and Additional Reading

TECHNISCHE UNIVERSITÄT DARMSTADT

- Slides will be published as the course proceeds
  - These serve as mere guidelines
- There is no single recommended textbook. Rather, multiple textbooks and papers.
  - Make sure to read the respective book chapters and papers

- Some Recommended Textbooks:
  - George Coulouris, Jean Dollimore, and Tim Kindberg. 2005. **Distributed Systems: Concepts and Design.** Addison-Wesley Longman.
  - Andrew S. Tanenbaum and Maarten Van Steen. 2001. **Distributed Systems: Principles and Paradigms.** Prentice Hall.
  - M. Tamer Özsu and Patrick Valduriez. 2011. **Principles of Distributed Database Systems** (3nd Ed.). Prentice-Hall.
  - Jim Gray and Andreas Reuter. 1992. **Transaction Processing: Concepts and Techniques**. Morgan Kaufmann.
  - Shahram Dustdar Harald Gall Manfred Hauswirth **Software-Architekturen für Verteilte Systeme**. Springer Verlag 2003
  - Len Bass, Paul Clements, Rick Kazman **Software Architecture in Practice (2nd Edition).** Addison Wesley. 2005
  - Gregor Hohpe and Bobby Woolf. **Enterprise Integration Patterns**: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley 2003
  - Qusay Mahmoud **Middleware for communications**. John Wiley and Sons, 2004
  - … various chapters form other books

Reading for THIS Lecture

TECHNISCHE UNIVERSITÄT DARMSTADT

- The slides for the lecture are based on material from:
  - George Coulouris, Jean Dollimore, and Tim Kindberg. 2005. **Distributed Systems: Concepts and Design.** Addison-Wesley Longman.
    - Chapter 2
  - Andrew S. Tanenbaum and Maarten Van Steen. 2001. **Distributed Systems: Principles and Paradigms.** Prentice Hall.
    - Chapter 1
  - P.A. Bernstein. **Middleware**. CACM, Feb. 1996/Vol.39 No.2
  - Jim Gray, Prashant Shenoy. **Rules of Thumb in Data Engineering**. IEEE International Conference on Data Engineering, Feb 28-30, San Diego, CA. 2000
  - David A. Patterson. 2004. **Latency lags bandwith**. Commun. ACM 47, 10 (October 2004), 71-75.

Definitions and Challenges

TECHNISCHE UNIVERSITÄT DARMSTADT

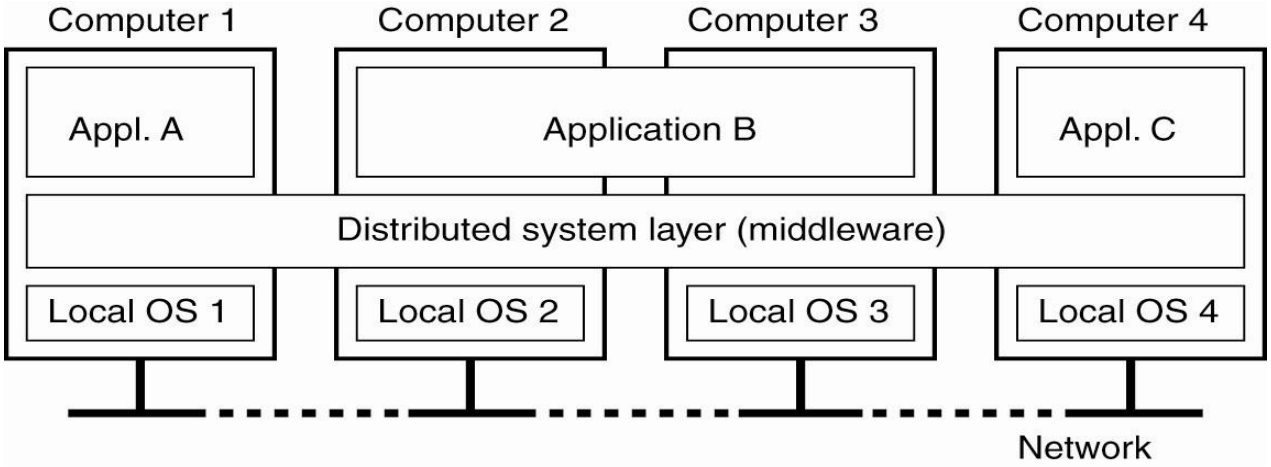
Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Definition of a Distributed System

TECHNISCHE UNIVERSITÄT DARMSTADT

- A **distributed system** is a collection of autonomous computers linked by a network and equipped with a distributed software system
  - It appears to its users as a single coherent system



- A **middleware** layer:
  - spreads over multiple machines
  - offers each application the same interface
  - abstracts underlying details, distribution and communication → transparency

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms

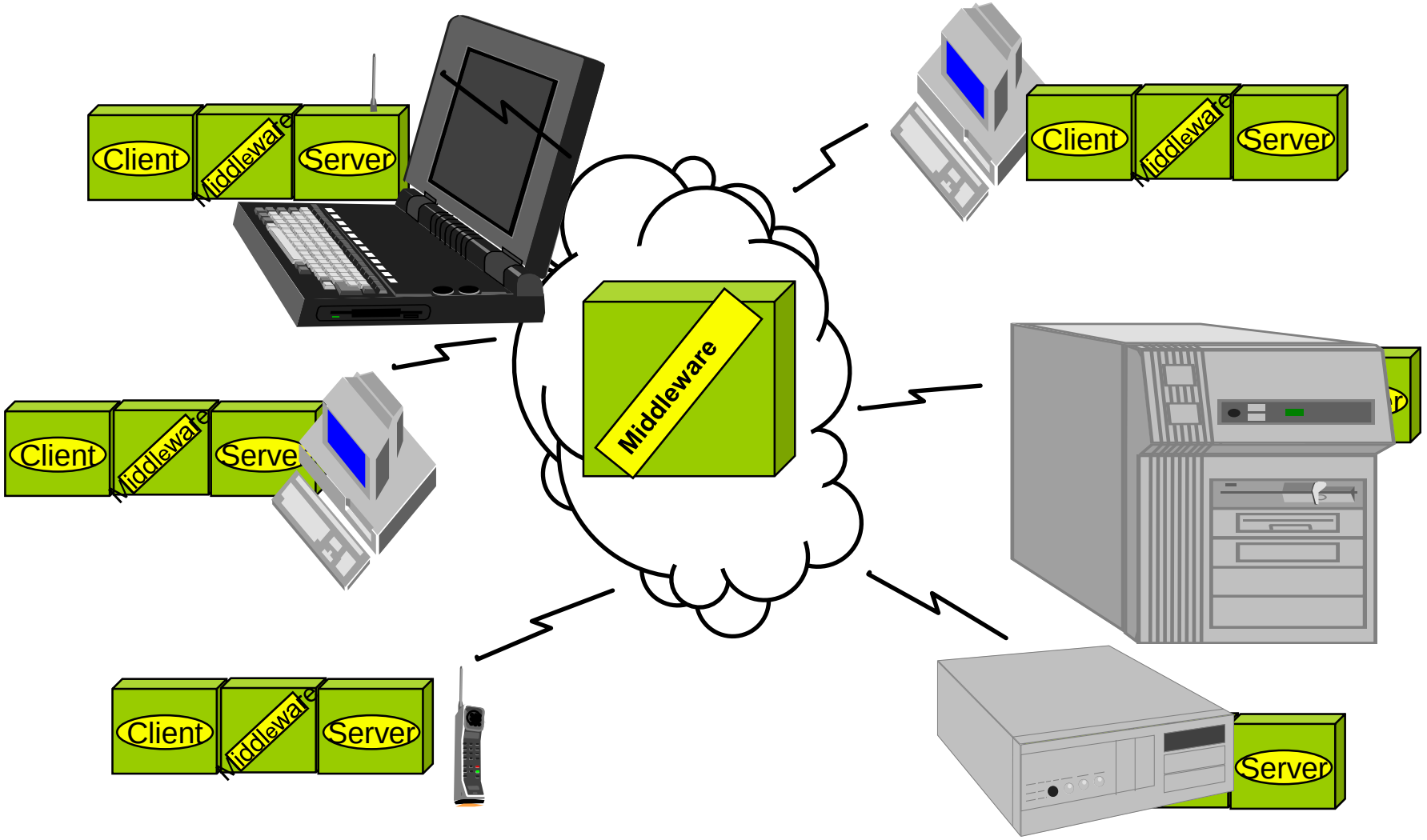
DVS

Aspects of the definition

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hardware
  - Autonomous machines
  - Different hardware
  - Failures
  - → Developers view X machines as a single one, with X-times as many resources
- Software
  - Development – requires new abstraction and poses new problems
  - Communication and reliability – bring new aspects unknown from single machine systems (Independent failures, coordination)
  - Time - no global clocks
  - Consistent state of software and data – new dimensions
  - Layered architecture – transparency, ease of development
- Resources: hardware, software entities, data;
  - Identifiable (naming scheme), accessible (naming to address mapping), operations (management policies)

Example

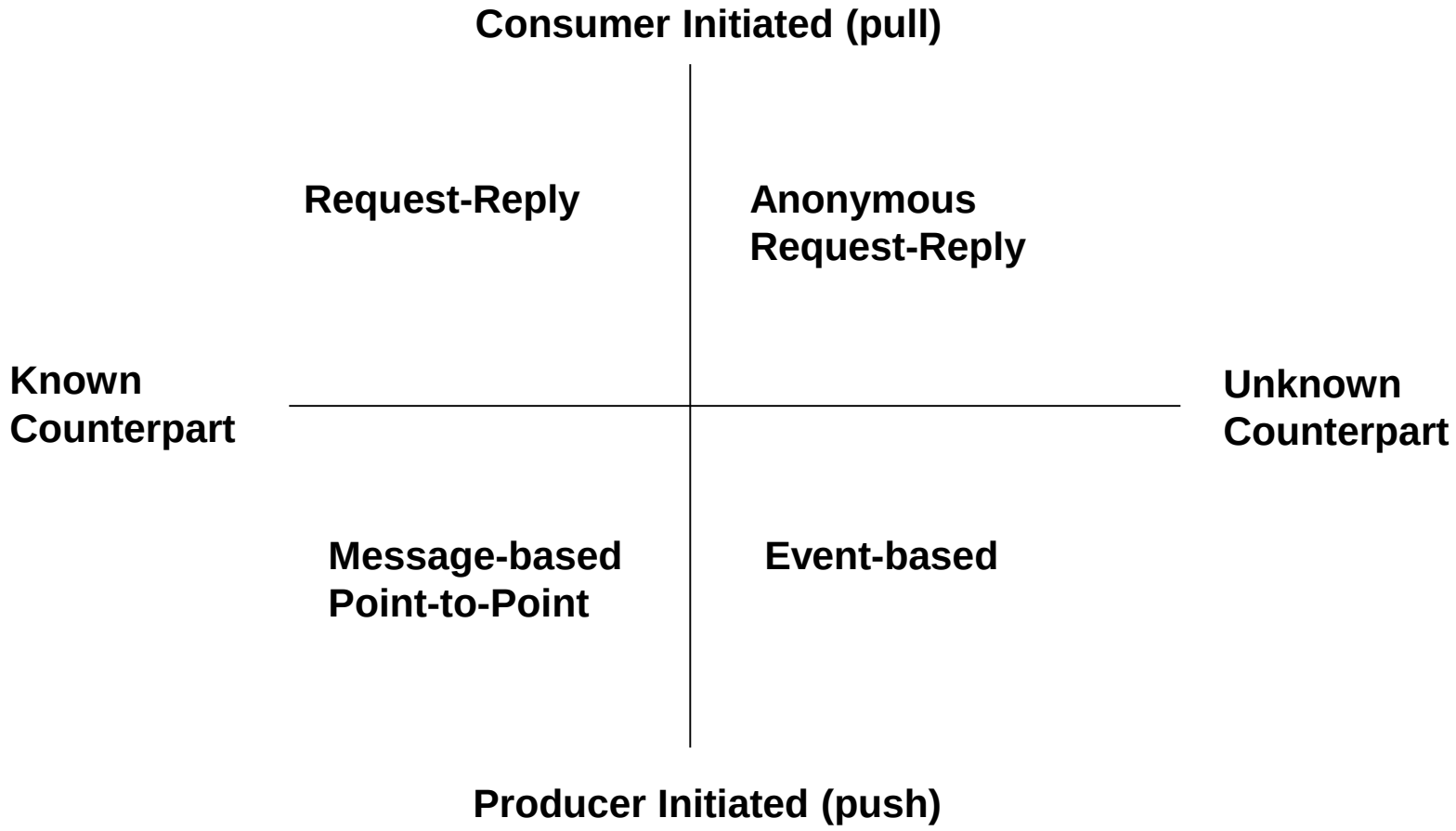

Client
Middleware
Server
Client
Middleware
Server
Middleware
Client
Middleware
Server
Client
Middleware
Server
Server

Challenges

TECHNISCHE UNIVERSITÄT DARMSTADT

- Invocation
- Communication
- Transparency
- Heterogeneity
- Scalability
- Consistency (data)
- Synchronization (processes)
- High Availability
- Fault Tolerance
- Openness
- Mobility
- *Security – covered in other courses*

Interactions



Communication

TECHNISCHE UNIVERSITÄT DARMSTADT

- Interactions are implemented through communication among processes
- Networking infrastructure
  - Interconnections | network software
- Communication primitives and models and their implementation:
  - communication primitives:
    - {send | receive} → message passing
    - remote procedure call (RPC) | Async. RPC | TRPC
  - communication models
    - Client-Server communication: implies a message exchange between two processes: the process which requests a service and the one providing it
    - Unicast: message targets one process
    - Group multicast: the target of a message is a set of processes, which are members of a given group
    - Publish/Subscribe: subscriber subscribes to events produced by a producer. Producer does not know consumer, middleware mediates

Transparency

TECHNISCHE UNIVERSITÄT DARMSTADT

- Transparency is essential principle in middleware → hiding of details
  - Many implications: Interface Descr. Languages, Obj.Ref. (IDs), Marshalling

Transparency	Description
Access	Hide differences in data representation and how a resource (software, device, network) is accessed, or is available
Location	Hide details about physical resource location → naming
Migration/ Relocation	Hide that a resource may move to another location (while in use)
Replication	Hide the fact that several copies exist (→ location transparency)
Concurrency/Tx	Hide that a resource may be shared by several competitive users. Resource: Object, Process, Data Entity, Device. Method: Locks/semaphores/transactions…
Failure	Hide (mask) the failure and recovery of a resource. Example: What is a failure: (i) server down or (ii) server overloaded?
Persistence	Hide how and where a resource (data) is persisted

Degree of Transparency

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hiding all details (full transparency) → unrealistic
  - Implication: performance and complexity

- Full transparency costs performance, complexity and still exposes the distribution of the system

Scalability

TECHNISCHE UNIVERSITÄT DARMSTADT

- Several aspects with respect to :
  - Size – add users to the system
  - Performance and resource utilization – more resources vs. speed-up
  - Geographical distribution – distribute a system on three continents
    - Synchronous vs. asynchronous communication, Latency, Consistency
  - Maintenance and administration

- Scalability problems: centralization vs. distribution
  - Centralized software and algorithms
    - Simpler but does not scale
  - Distributed software and algorithms
    - Scale well but much more complex, depend on assumptions made
      - No complete information about global state
      - No global clock assumption
      - Decisions based on local information
      - Consistency conditions may have to be relaxed
      - Local failures do not affect the algorithm

Scalability - Solutions

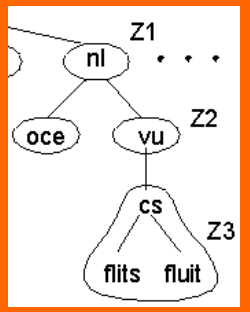
TECHNISCHE UNIVERSITÄT DARMSTADT

- Hide Latencies
  - Reduce response and waiting times – avoid blocking
  - Asynchronous communication
  - Reduce Roundtrips: bulk operations or parallelism
- Distribution
  - Decompose a system
  - Partition Logic and Data across nodes
- Caching and Replication
  - Logic, Data
  - Better Performance, Load balancing, Availability
  - Consistency Problems → synchronization cost
    - Can inconsistencies be tolerated?
- Parallelism

Example - Latencies:

- Filling a form asynchronously with AJAX
- Filling a form field by field and validating it on server
- Providing all field data at once, validated at client, constraints shipped

Example Distribution - DNS:



Example Caching:

- **Speedup processing**
- **Cache eviction**

Concurrency/Transactions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Managing shared resources
  - Data, State
  - Request at a time vs. concurrent requests
  - Resource utilization and system availability

- System should provide primitives to make an object safe in a concurrent environment

- Required:
  - Atomicity
  - Consistency
  - Isolation
  - Durability (ACID)

→ Transactions, transactional models, system support

CAP Theorem

TECHNISCHE UNIVERSITÄT DARMSTADT

Brewer's CAP theorem states:

In a distributed computer system you can only guarantee two of the following three properties simultaneously:

- Consistency
- Availability
- Partition tolerance

Brewer's conjecture (PODC 2000) was proven later by Nancy Lynch and Seth Gilbert making more narrow assumptions.

Correctness moves from ACID to BASE (Basic Availability, Soft state, Eventual consistency)

Failure handling and tolerance

TECHNISCHE UNIVERSITÄT DARMSTADT

- Faults occur in: hardware, software, *communication*. What is a fault?
- Processes may:
  - produce incorrect/arbitrary results
  - stop before completing an operation
  - not respond due to overload or communication failure

- Techniques for dealing with failures:
  - Detecting failures
    - detectable (checksums), undetectable (availability of server), random
    - manage failures that cannot be detected but are suspected
  - Masking failures:
    - Attempt Repair – resend message, rebuild disc arrays
    - Tolerating failures – acknowledge failure, run the rest of the logic
    - Recovering from failures – recover after crashes, restore consistent state
  - Redundancy – tolerate failures by using redundant components, hot standby, failover
- High Availability

Security

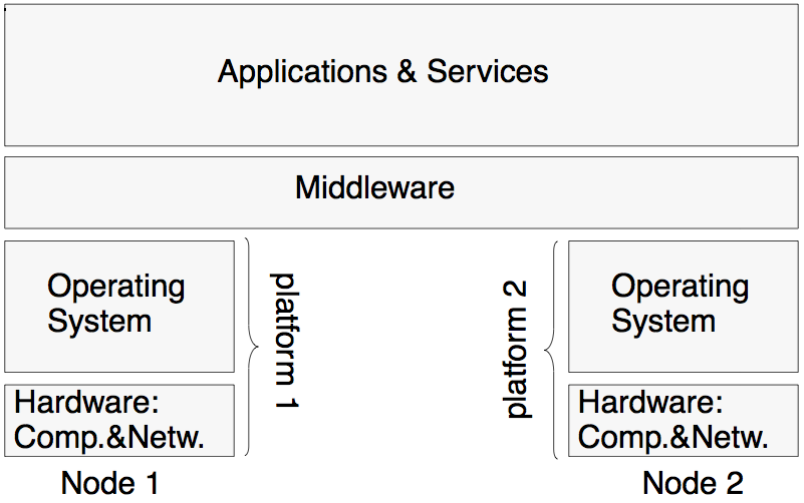
TECHNISCHE UNIVERSITÄT DARMSTADT

- Security has three components:
  - Confidentiality: protection against disclosure to unauthorized individuals
  - Integrity: protection against alteration or corruption
  - Availability: protection against interference with the means to access the resources

- Some typical security challenges in large distributed systems:
  - Denial of service attacks (DoS)
  - Security of mobile code

Openness

- Open systems are characterized by the fact that their key interfaces are published and designed according to the rules of standard syntax and semantics
- Goal: Interoperability and Portability
- Interface description:
  - Published and discoverable
  - Interface Description Languages
- Standard protocols:
  - format, contents, meaning of messages
- Specifications are complete and neutral:
  - Implementation independent, and completely specified



Heterogeneity

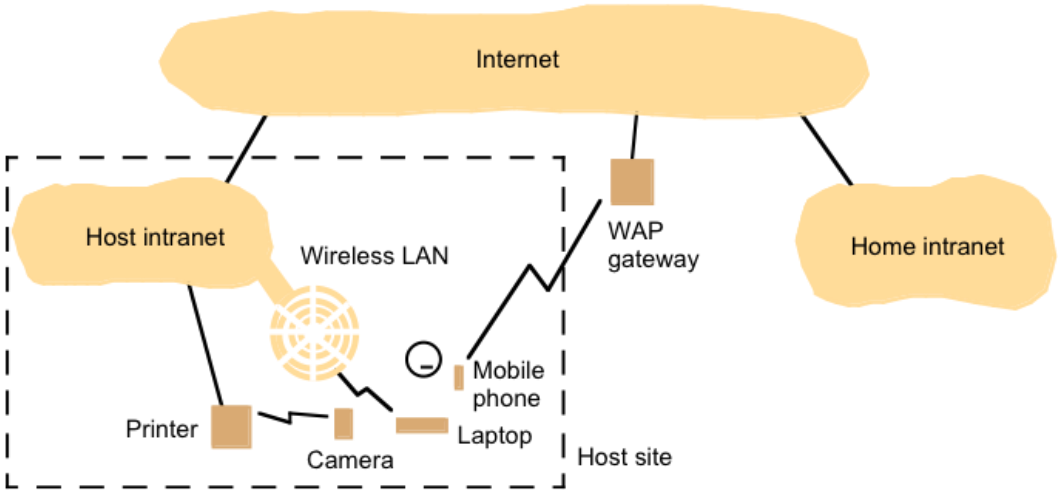
- Many aspects
  - Hardware, software
  - Data formats and representation, Types and schemata (semantic heterogeneity)
  - Networks: Interfaces, Protocols
  - Programming languages
  - Systems … !

- Middleware:
  - provides programming abstractions, masking the heterogeneity,
  - under manageable complexity and performance loss

Mobility

- Mobile and nomadic users
  - Connection interrupts
  - Link Asymmetry

- Consistency
- Replication
- Synchronization



Distribution

TECHNISCHE UNIVERSITÄT DARMSTADT

DVS

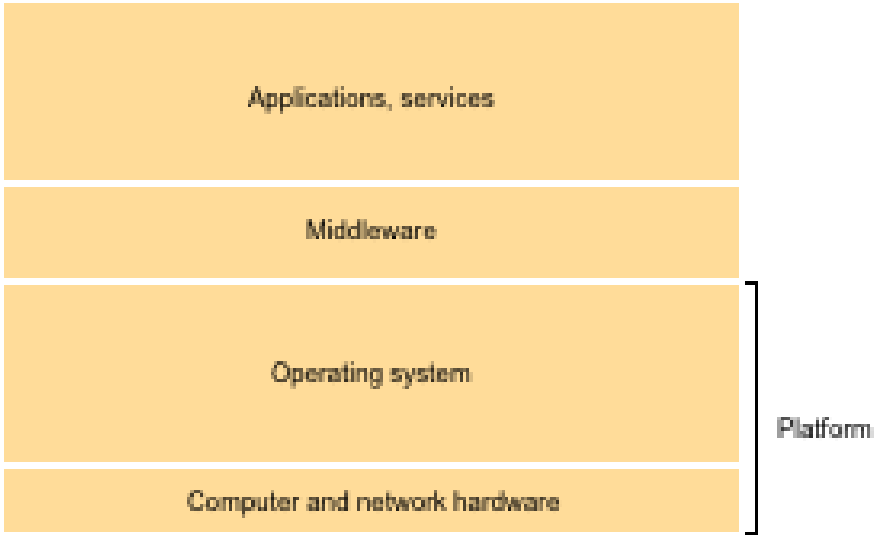
Software Layers

TECHNISCHE UNIVERSITÄT DARMSTADT

- Software architecture of distributed systems comprises many layers

- Application - comprises one or more clients consuming a service
- Server – offers a service, accepts and processes requests
- Distributed service - provided by more than one interacting processes
- Platform
  - The lowest layer
  - Hardware
  - Base Software (OS, VM, ..)

- Middleware (see Challenges)
  - Abstract
  - Offer programming model
  - Offer a set of services

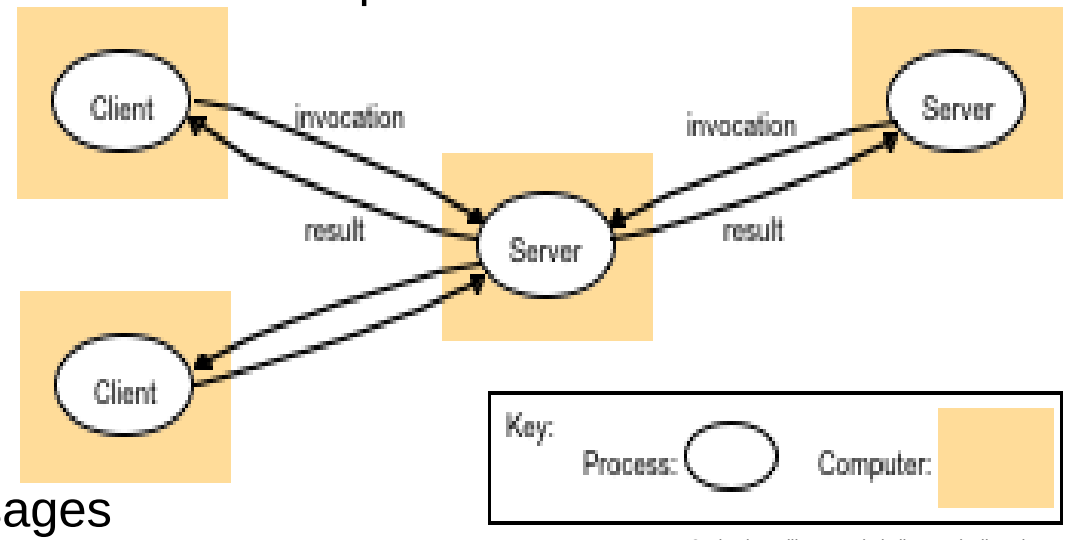


Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design

DVS

Client/Server

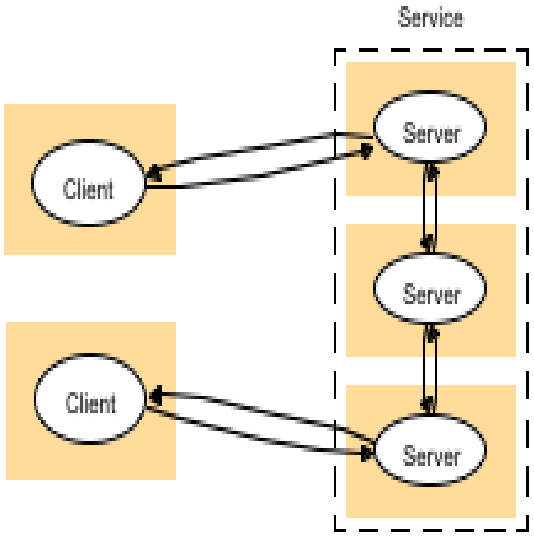
- Clients and Servers are logical entities that cooperate in order to carry out a certain task
- Client and Server → Client requests service from server (transitive roles) and (depending on implementation) blocks until response arrives
  - Logically separated!
  - Physically Separated: Address space, Time, Computing Site?
- Service
  - The server provides services
  - The client uses/consumes them
  - Interface → ecapsulation
- Operations are based on messages
- Encapsulation of services
  - As long as the interface is maintained its implementation can be replaced
- Bottom line: Simplicity, poor Scalability, Centralization



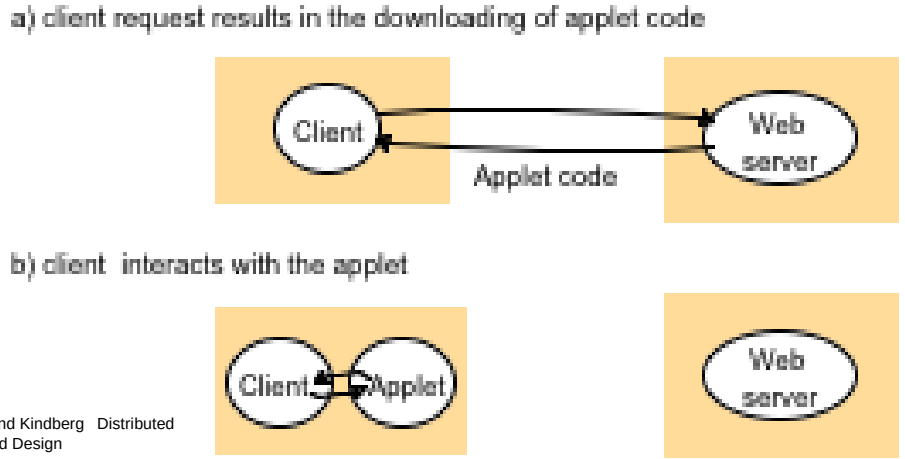
Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design

Client/Server evolution

- Service → multiple cooperating servers on several hosts
- Replicated distributed services
- Examples
  - Web Servers
  - NIS (Network Information Service)
  - Cluster architectures



- Mobile code
  - Push code to the client
  - Local processing
  - Mask latency, Scalability
- Example
  - Fat client vs. fat server
  - Java Script, Java Applets



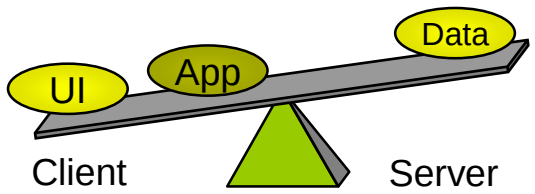
Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design

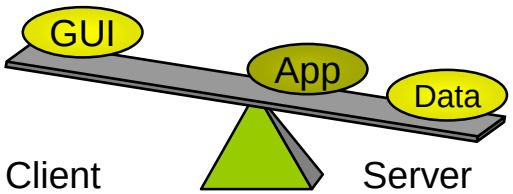
Fat Client vs. Fat Server (Example)

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Fat Client (Functionality at client)**
- Integrity and Consistency (requires data at client)
  - Data type verification, ranges, etc. – integrity
    - e.g. Date of birth verification
  - Verify existence of data (referential integrity)
  - Calculate the next employee number [Autonumber] (read last; add one; return it)
    - e.g. Employee number (unique)

- **Fat Server (Functionality at Server)**
- Integrity and Consistency
  - Based on Stored Procedures
  - Centrally controlled referential integrity
  - A unique version of the verification process is maintained at the server side
- Autonumber is a Stored Procedure that controls uniqueness





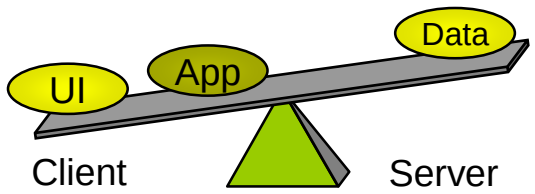
DVS

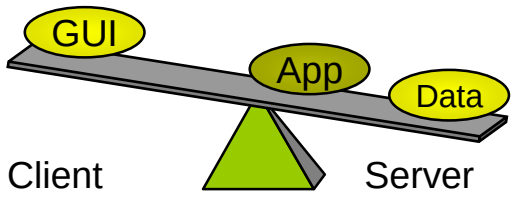
Fat Client vs. Fat Server - Example

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Fat Client**
- Business Rules-interchangeable!
  - They could be application-specific
  - Business rules scattered in many applications
  - Expressed in form of programming code
- Example:
  - Department Bonus (10% for all programmers)

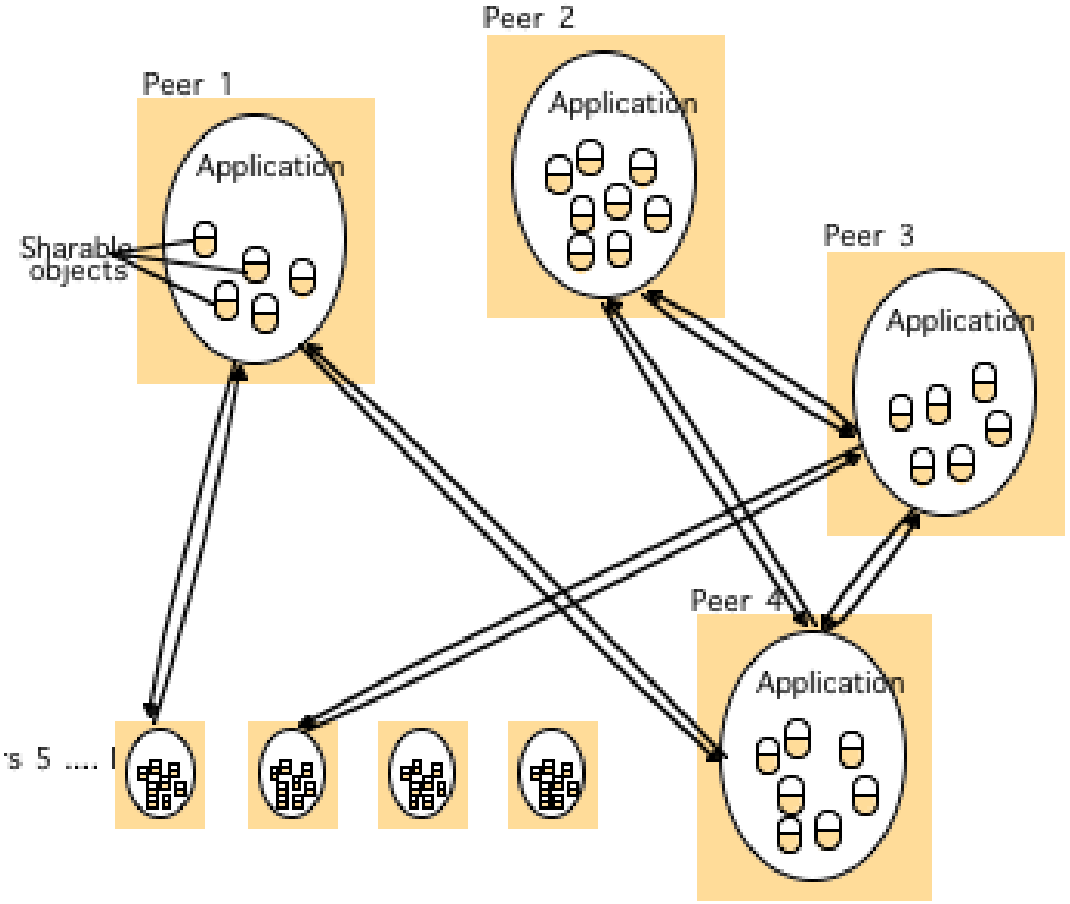
- **Fat Server**
- Business Rules
  - Stored Procedures & Triggers
  - Business rules are located in one place (the server)
  - Example:
    - Department Bonus (10% for all programmers)
    - Employee table (column dept) has a trigger that is executed on update or on insert. The trigger in fact executes a stored procedure





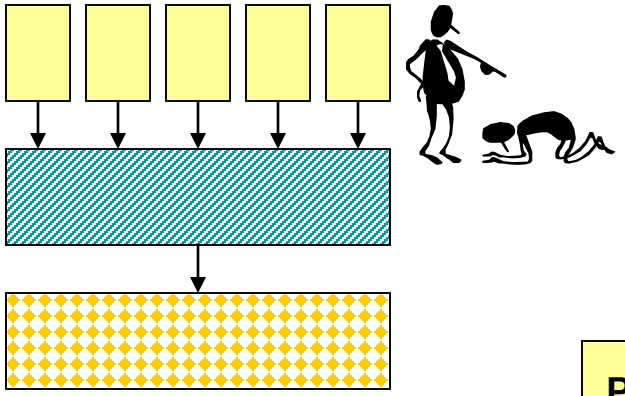
DVS

Peer-to-Peer

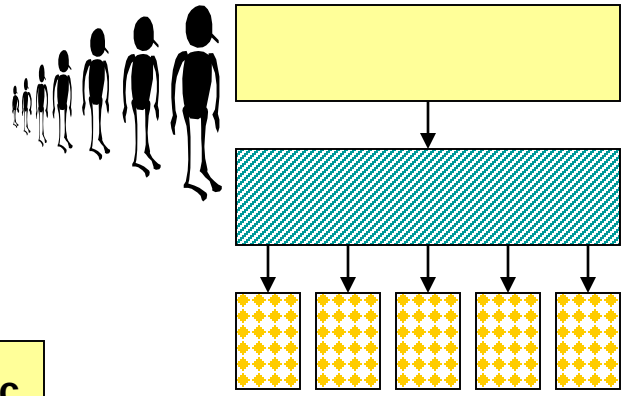


- Architecture comprises
  - Set of cooperative peers
  - (no client/server distinction)
  - Large number of peers
    - 100s …1 000 000s
  - Communication complexity
  - Replicated objects
  - Complex application development
- Bottom Line:
  - Better resource utilization
  - Resilience, Fault tolerance
  - Scalability
  - Complexity

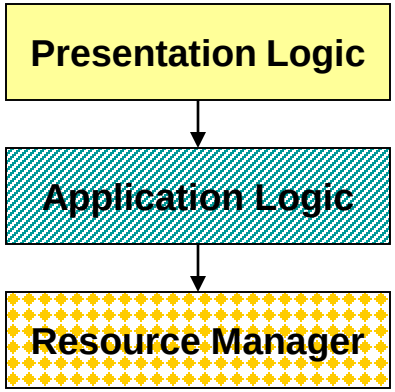
Distribution at the different layers



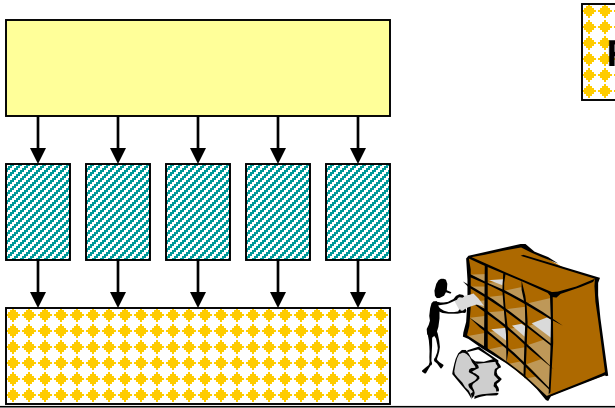
Support for multiple clients



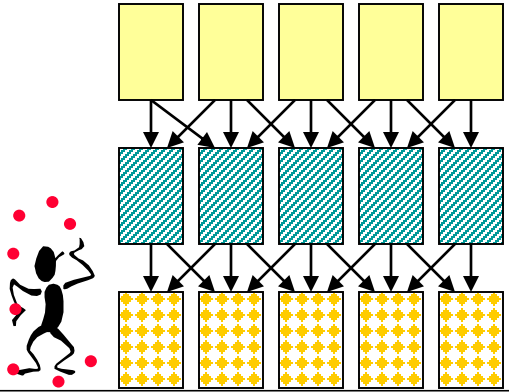
Data distribution or replication



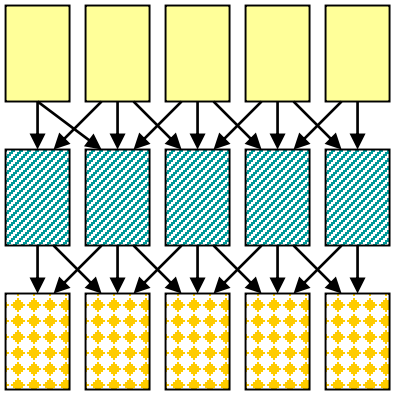
Separated application logic



Any combination thereof

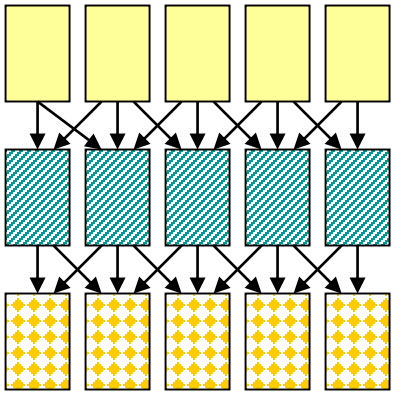


A game of boxes and arrows



- Each box represents a part of the system.
- Each arrow represents a connection between two parts of the system.
- The more boxes, the more modular the system: more opportunities for distribution and parallelism. This allows encapsulation, component based design, reuse.
- The more boxes, the more arrows: more sessions (connections) need to be maintained, more coordination is necessary. The system becomes more complex to monitor and manage.
- The more boxes, the greater the number of context switches and intermediate steps to go through before one gets to the data. Performance suffers considerably.
- System designers try to balance the capacity of the computers involved and the advantages and disadvantages of multiple layers.

A game of boxes and arrows



There is no problem in system design that cannot be solved by adding a level of indirection. There is no performance problem that cannot be improved by removing a level of indirection.

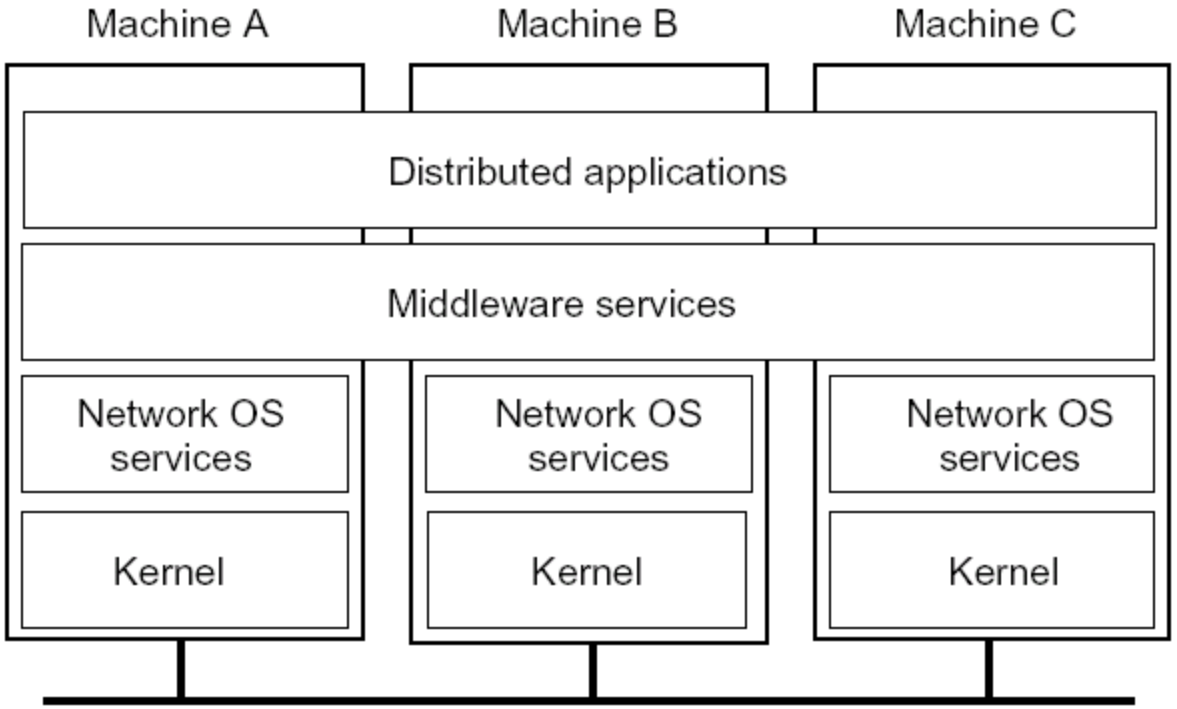
- Each box represents a part of the system.
- Each arrow represents a connection between two parts of the system.
- The more boxes, the more modular the system: more opportunities for distribution and parallelism. This allows encapsulation, component based design, reuse.
- The more boxes, the more arrows: more sessions (connections) need to be maintained, more coordination is necessary. The system becomes more complex to monitor and manage.
- The more boxes, the greater the number of context switches and intermediate steps to go through before one gets to the data. Performance suffers considerably.
- System designers try to balance the capacity of the computers involved and the advantages and disadvantages of multiple layers.

Middleware

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

Middleware

- Each computer/node is independent
- OS on different computers need not generally be the same
- Services are generally (transparently) distributed across computers



Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms

Middleware Services (1/2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Communication services**: abandon socket-based communication
  - Procedure calls across networks
  - Remote-object method invocation
  - Message-queuing systems
  - Event notification service

- **Information system services**: Services that help manage data in a distributed system
  - Large-scale, system-wide naming services
  - Advanced directory services
  - Location services for tracking mobile objects
  - Persistent storage facilities
  - Data caching and replication

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms

Middleware Services (2/2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Control services**: Services giving applications control over when, where, and how they access data
  - Distributed transaction processing
  - Code migration

- **Security services**: Services for secure processing and communication
  - Authentication and authorization services
  - Simple encryption services
  - Auditing service

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms

Fundamental Models

TECHNISCHE UNIVERSITÄT DARMSTADT

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Fundamental models

TECHNISCHE UNIVERSITÄT DARMSTADT

- Interaction model
  - Processes interact by passing messages:
  - Communication – latency, accuracy, reliability
  - Coordination – synchronization, ordering
- Failure model
  - Determines the type of fault (crash, intermittent, byzantine, …)
  - Goal: Preserve correct operation in presence of faults
  - Design to tolerate fault classes
- Security model

Interaction Model in Distributed Systems

TECHNISCHE UNIVERSITÄT DARMSTADT

- Processes have private local state (persistent data, program variables)
  - Changed only through interaction
  - No global state

1. Performance of communication channels
   - Latency – time to transmit data to destination
     - Physical limit – speed of light
   - Bandwidth – data volume transmitted over a unit of time (throughput)
     - Engineering issue (see rules of thumb)
   - Jitter – variation of time taken to deliver messages in a sequence of messages
   - Message transfer time = latency + length/data transfer rate

Reading: David A. Patterson. Latency lags bandwith. CACM 47, 10 (10, 2004), 71-75.

2. Computer clocks and timing events
   - Impossible to maintain a single global clock because of local clocks, clock drift rate

Interaction Model – 2

TECHNISCHE UNIVERSITÄT DARMSTADT

Time bounded interaction models ~~(sometimes called synchronous*)~~
- The time to execute an operation has known lower and upper bounds.
- Messages are received within a known bounded time
- Clock drift rates have a known bound

Unbounded interaction models ~~(sometimes called asynchronous)~~
- No bound on process execution speeds
- No bound on message transmission delays
- No bound on clock drift rates

- Time bounded systems can be built – very expensive, Real-Time Apps
  - Needed: Sufficient computing resources and network capacity
  - Timestamps and global physical time required
  - Networks with bounded delay

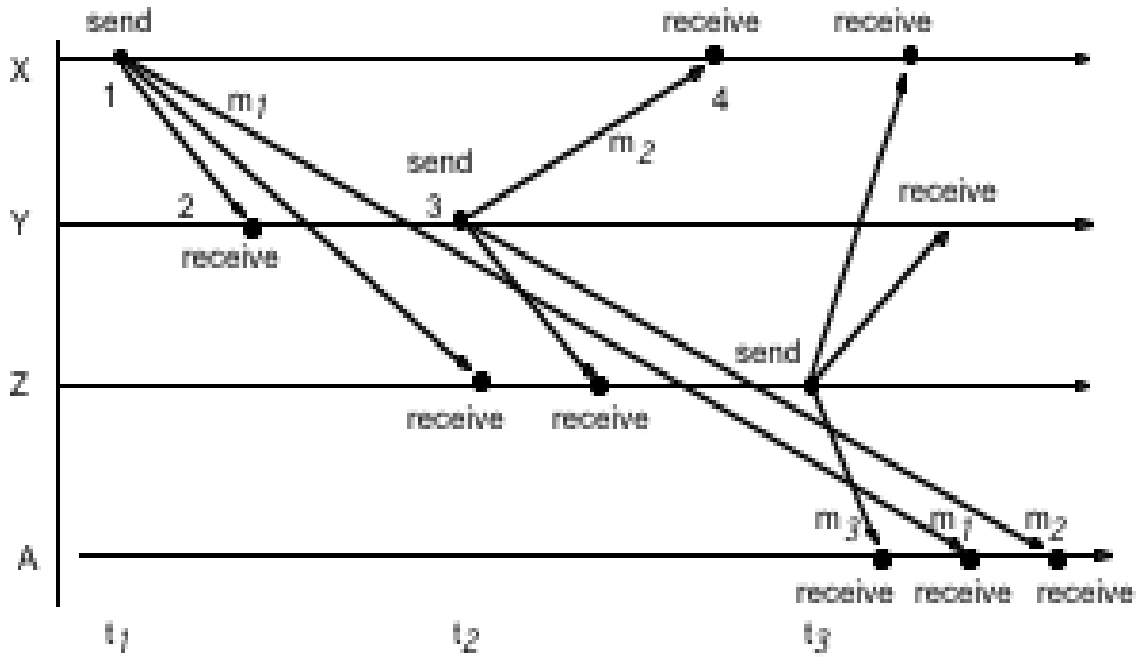
Actual distributed systems have unbounded delays

*Note: synchronous/asynchronous are overloaded terms, often used to mean blocking/non-blocking interactions

Interaction Model – Ordering

TECHNISCHE UNIVERSITÄT DARMSTADT

3. Event ordering
   - Event = sending | receiving a message
   - Example:
     - X,Y,Z, A – computers
     - X sends a message to {Y,A,Z}
     - Y,Z – Reply
   - Known
     - X sends m1 before Y receives m1
     - Y sends m2 before X receives m2
     - Y receives m1 before sending m2
   - Logical clocks:
     - Assign increasing num.
     - Relative order



DVS

Failure Model

TECHNISCHE UNIVERSITÄT DARMSTADT

- Omission failures
  - A Process or comm. Channel does not perform expected actions
  - Process → crash cleanly (functions correctly or stops)
  - Communication → drops messages → lack of buffer space or network error
- Arbitrary (Byzantine) failures
  - Worst case scenario: any erroneous behavior is possible
    - return none or wrong data or do not return at all (omission failure)
- Timing failures
  - Intermittent failures (often due to overload or synchronization problems)

Rules of Thumb

TECHNISCHE UNIVERSITÄT DARMSTADT

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Technology trends matter

TECHNISCHE UNIVERSITÄT DARMSTADT

- Network performance = message transmission rate
  - **Delay = latency + TransmissionTime = latency + msg_size/bandwidth**
  - **MessageCost (time) = senderCPU + receiverCPU + TransmissionTime**

	Example	*Range*	*Bandwidth (Mbps)*	*Latency (ms)*
Wired:				
LAN	Ethernet	1-2 kms	10-10000	1-10
WAN	IP routing	worldwide	0.010-600	100-500
MAN	ATM	250 kms	1-150	10
Internetwork	Internet	worldwide	0.5-600	100-500
Wireless:				
WPAN	Bluetooth (802.15.1)	10 - 30m	0.5-2	5-20
WLAN	WiFi (IEEE 802.11)	0.15-1.5 km	2-54	5-20
WMAN	WiMAX (802.16)	550 km	1.5-20	5-20
WWAN	GSM, 3G phone nets	worldwide	0.01-0.2	100-500

Gilder's Law

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Deployed bandwidth triples every year*
  - 3x bandwidth/year (for 25 more years)
  - Aggregate bandwidth doubles every 8 months!

Reading: Jim Gray, Prashant Shenoy Rules of Thumb in Data Engineering. ICDE 2000.

- Latency bounded by the speed of light
  - 60ms – 200ms round trip within North America, Europe, Asia.
  - Use caching to quickly access remote data

- In the time that bandwidth doubles, latency improves 1.2x
  - Bandwidth improves at least as the square of the improvement in latency

Reading: David A. Patterson. Latency lags bandwith. CACM 47, 10 (10, 2004), 71-75.

- Disk Latency (for comparison)
  - On an HDD: 8ms | On an SSD 0.2 ms

- If an application can reuse data, should it cache it or re-fetch it?

Network IO vs. Disk IO

TECHNISCHE UNIVERSITÄT DARMSTADT

- A network message costs 10,000 instructions and 10 instructions/byte.
- A disk IO costs 5,000 instructions and 0.1 instructions /byte.
- Open question: What are the system implications?

Reading: Jim Gray, Prashant Shenoy Rules of Thumb in Data Engineering. ICDE 2000.

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Key notions
  - Distributed System
  - Middleware
- Key characteristics and challenges
- Distribution models
- Fundamental models
- Rules of thumb

Thank You!

TECHNISCHE UNIVERSITÄT DARMSTADT

Questions?

DVS