Systems Infrastructure for Data Science

Web Science Group
Uni Freiburg

WS 2014/15

Lecture VI: Introduction to Distributed Databases

Why do we distribute?

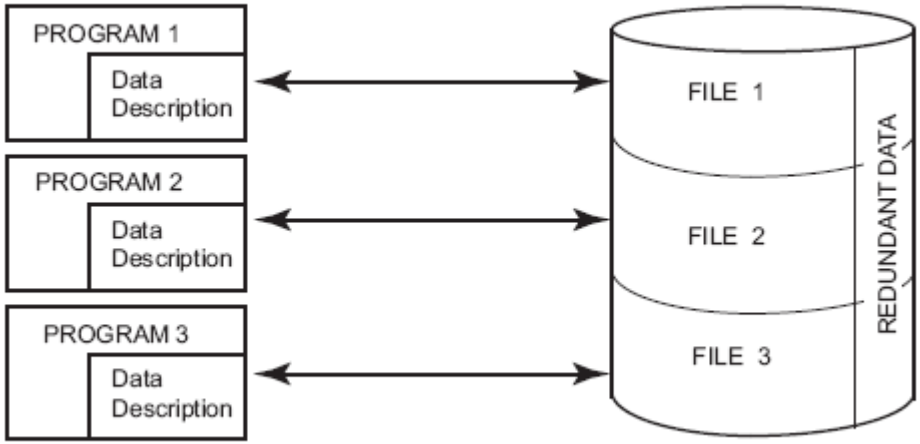
- Applications are inherently distributed.
- A distributed system is more reliable.
- A distributed system performs better.
- A distributed system scales better.

Distributed Database Systems

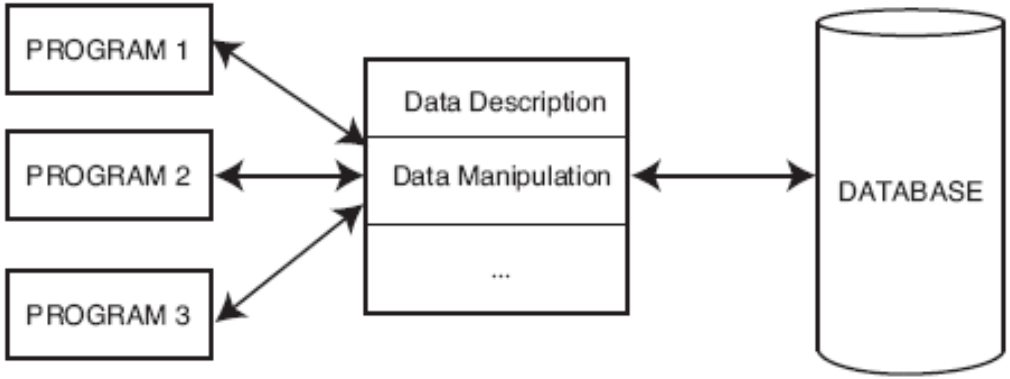
- Union of two technologies:
  - Database Systems + Computer Networks
- Database systems provide
  - data independence (physical & logical)
  - centralized and controlled data access
  - integration
- Computer networks provide distribution.
- integration ≠ centralization
- integration + distribution

DBMS Provides Data Independence

File Systems



Database Management Systems



Distributed Database Systems

- Union of two technologies:
  - Database Systems + Computer Networks
- Database systems provide
  - data independence (physical & logical)
  - centralized and controlled data access
  - integration
- Computer networks provide distribution.
- integration ≠ centralization
- integration + distribution

Distributed Systems

- Tanenbaum et al:

  "*a collection of <u>independent</u> computers that appears to its users as a <u>single coherent</u> system*"

- Coulouris et al:

  "*a system in which hardware and software components located at <u>networked computers</u> communicate and coordinate their actions only by <u>passing messages</u>*"

Distributed Systems

- Ozsu et al:

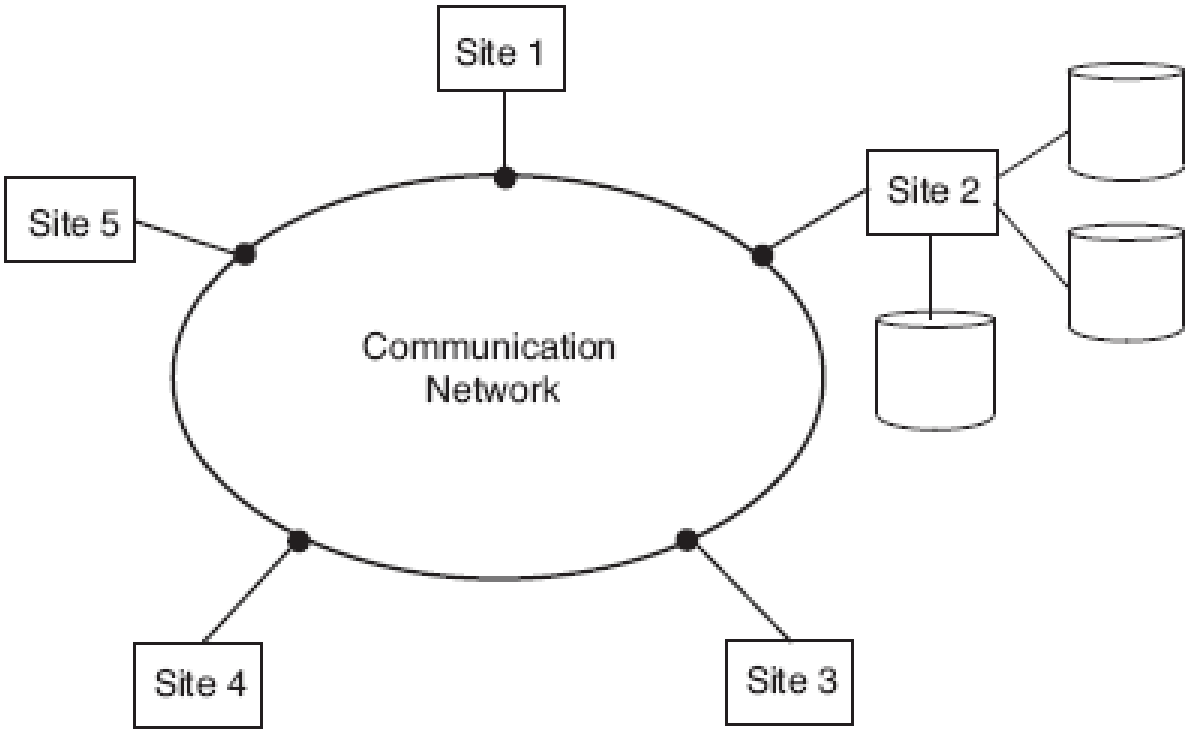
  "a number of autonomous processing elements (not necessarily homogeneous) that are interconnected by a computer network and that cooperate in performing their assigned tasks"

What is being distributed?

- Processing logic
- Function
- Data
- Control

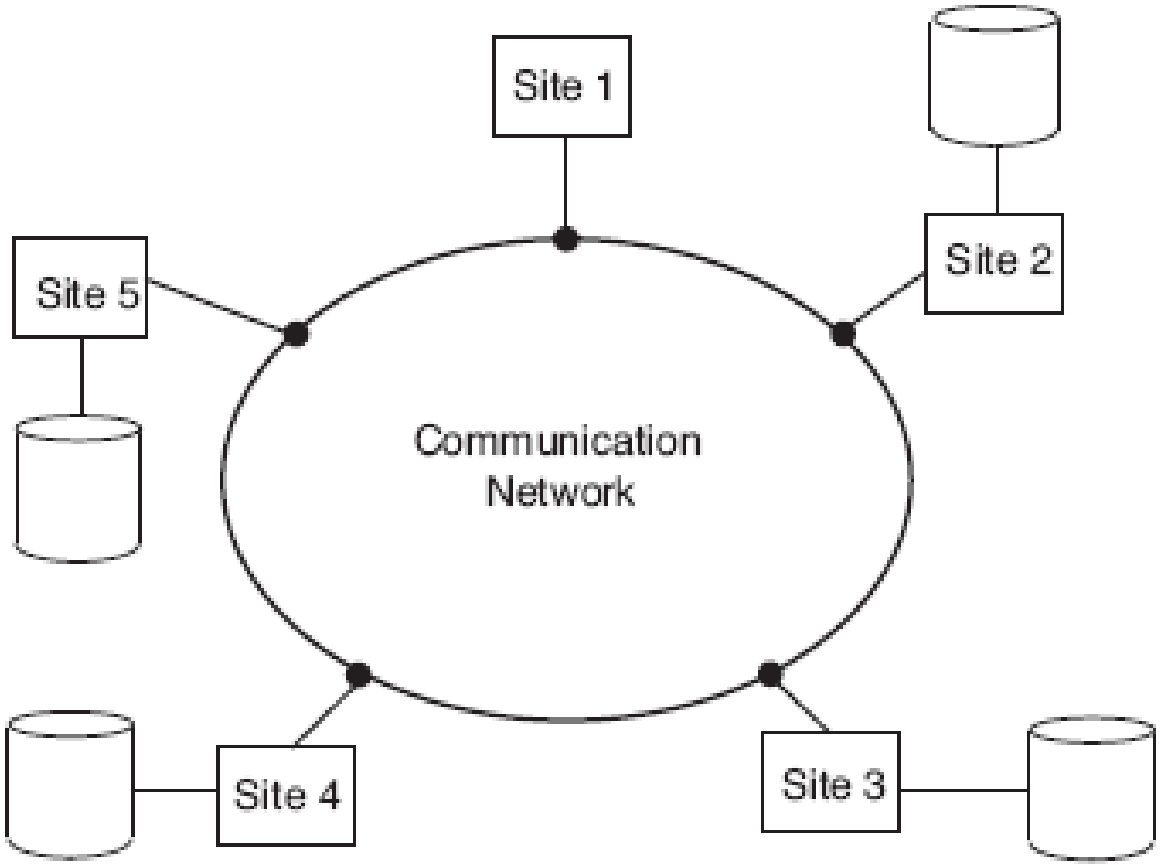
- For distributed DBMSs, all are required.

Centralized DBMS on a Network



What is being distributed here?

Distributed DBMS



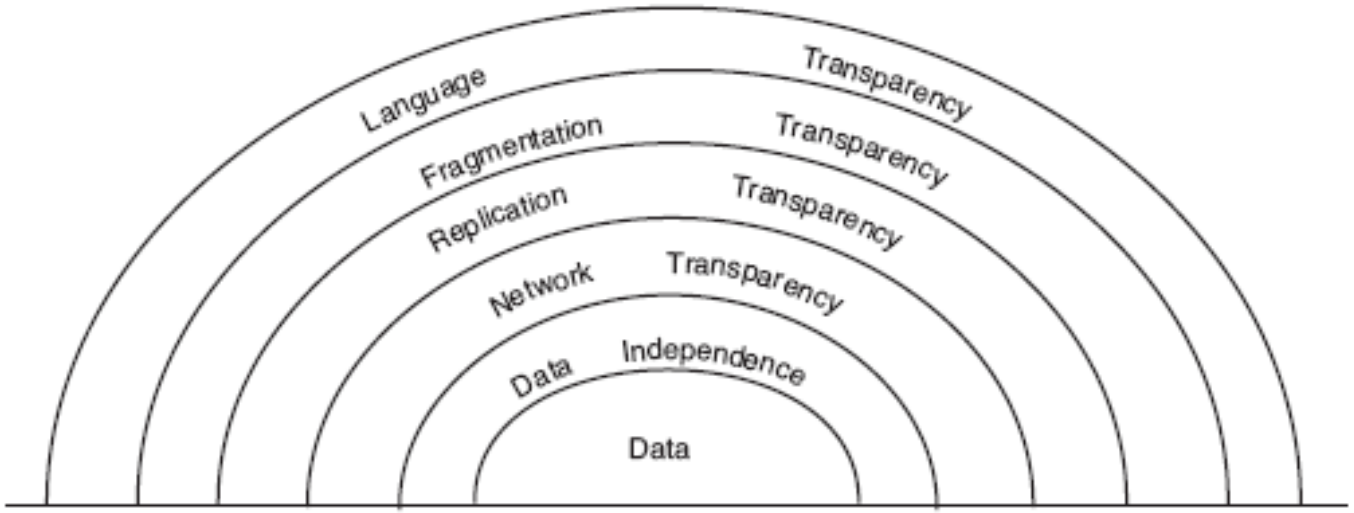
And here?

Distributed DBMS Promises

1. Transparent management of distributed and replicated data
2. Reliability/availability through distributed transactions
3. Improved performance
4. Easier and more economical system expansion

Promise #1: Transparency

- Hiding implementation details from users
- Providing **data independence** in the distributed environment
- Different transparency types, related:



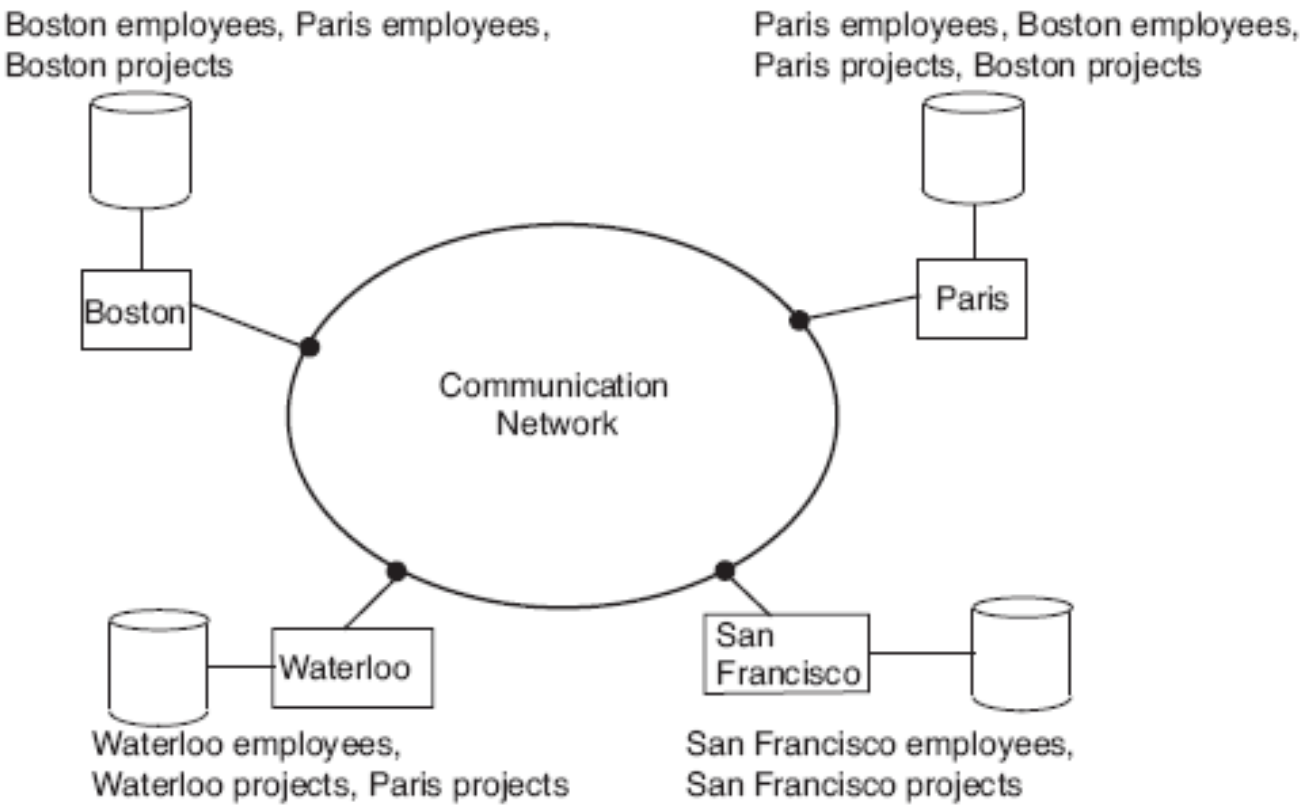
- Full transparency is neither always possible nor desirable!

Transparency Example

- Employee (eno, ename, title)
- Project (pno, pname, budget)
- Salary (title, amount)
- Assignment (eno, pno, responsibility, duration)

```
SELECT    ename, amount
FROM      Employee, Assignment, Salary
WHERE     Assigment.duration > 12
AND       Employee.eno = Assignment.eno
AND       Salary.title = Employee.title
```

Transparency Example



What types of transparencies are provided here?

Promise #2: Reliability & Availability

- Distribution of replicated components
- When sites or links between sites fail
  - No single point of failure
- Distributed transaction protocols keep database consistent via
  - Concurrency transparency
  - Failure atomicity
- Caveat: CAP theorem!

Promise #3: Improved Performance

- Place data fragments closer to their users
  - less contention for CPU and I/O at a given site
  - reduced remote access delay
- Exploit parallelism in execution
  - inter-query parallelism
  - intra-query parallelism

Promise #4: Easy Expansion

- It is easier to scale a distributed collection of smaller systems than one big centralized system.

Distributed DBMS Major Design Issues

- Distributed DB design (Data storage)
  - partition vs. replicate
  - full vs. partial replicas
  - optimal fragmentation and distribution is NP-hard
- Distributed metadata management
  - where to place directory data
- Distributed query processing
  - cost-efficient query execution over the network
  - query optimization is NP-hard

Distributed DBMS Techniques: Partitioning

- Each relation is divided into n partitions that are mapped onto different systems/locations.
- Provides storing large amounts of data and improved performance
- Fragmentation of tables:
  - Among rows/values
  - Among columns

Distributed DBMS Techniques : Replication

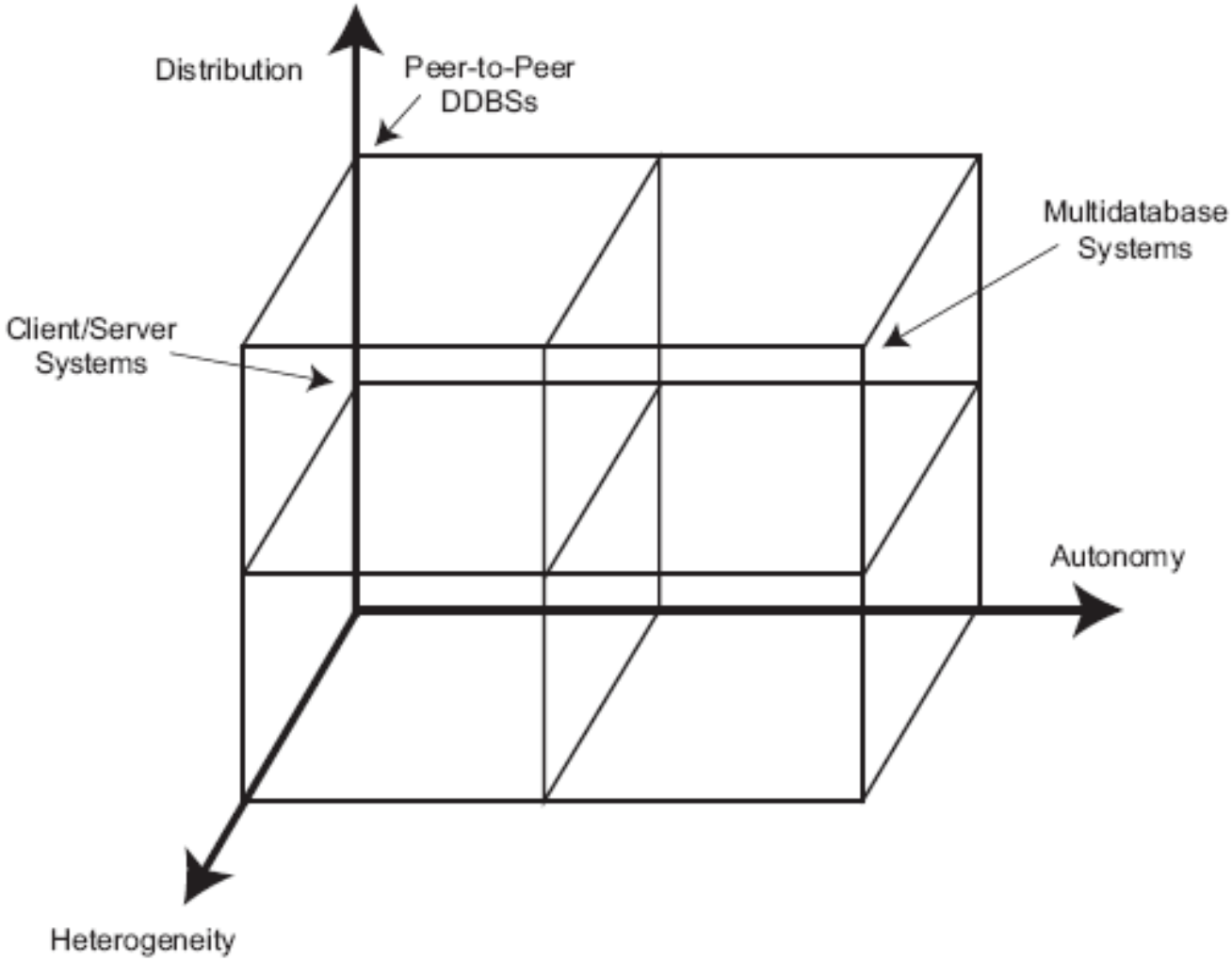
- Storing copies of data on different nodes
- Provides high availability and reliability
- Requires distributed transactions to keep replicas consistent:
  - Two phase commit - data always consistent but the system is fragile
  - Eventually consistency - eventually becomes consistent but always writable

Distributed transaction management

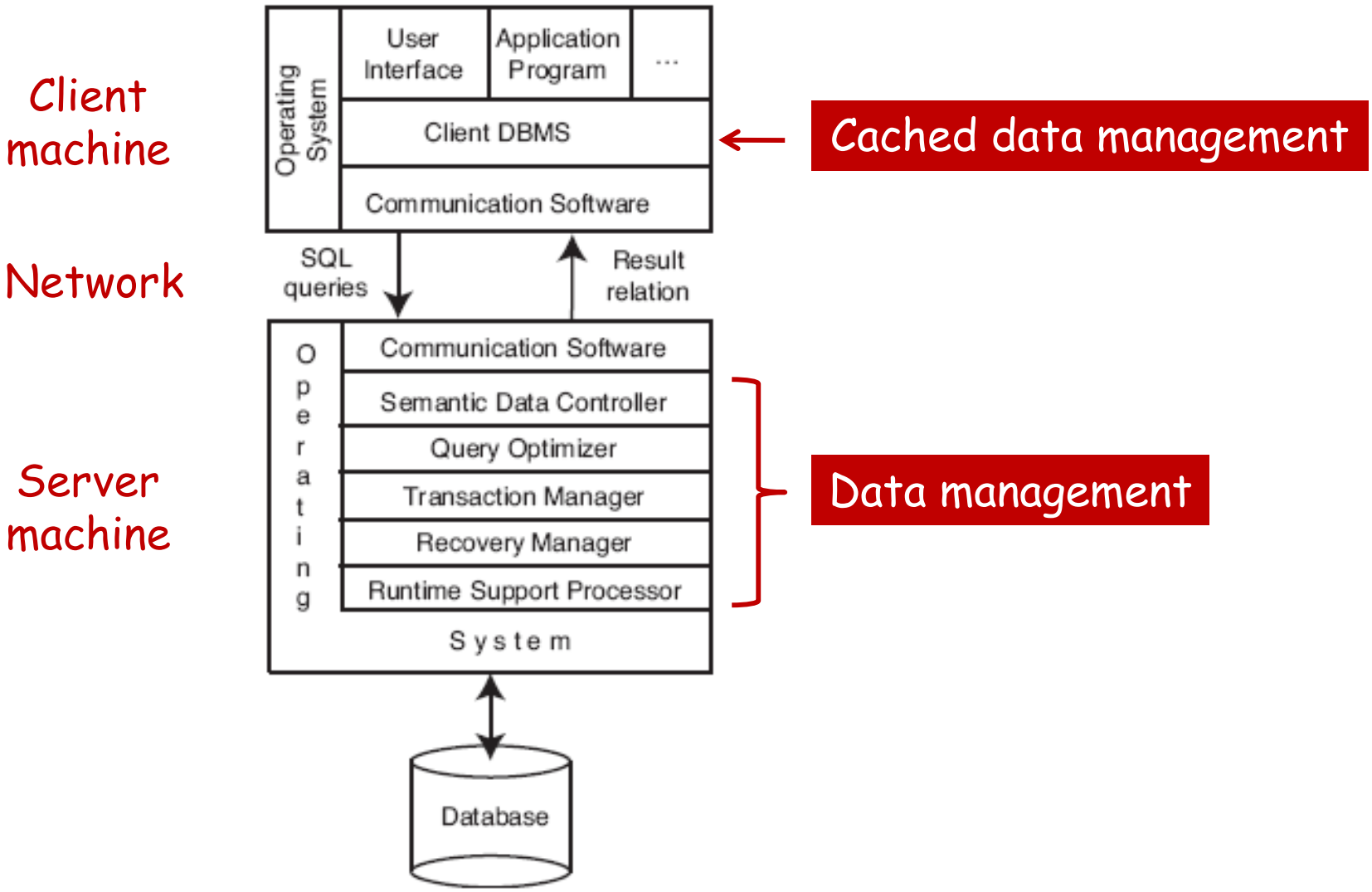
- Synchronizing concurrent access
- Consistency of multiple copies of data
- Detecting and recovering from failures
- Deadlock management
- Providing consistency (e.g., ACID properties)

=> Distributed Systems Lecture (w/ Prof. Schindelhauer in SS 2014, w/ Prof. Kuhn in SS2015)

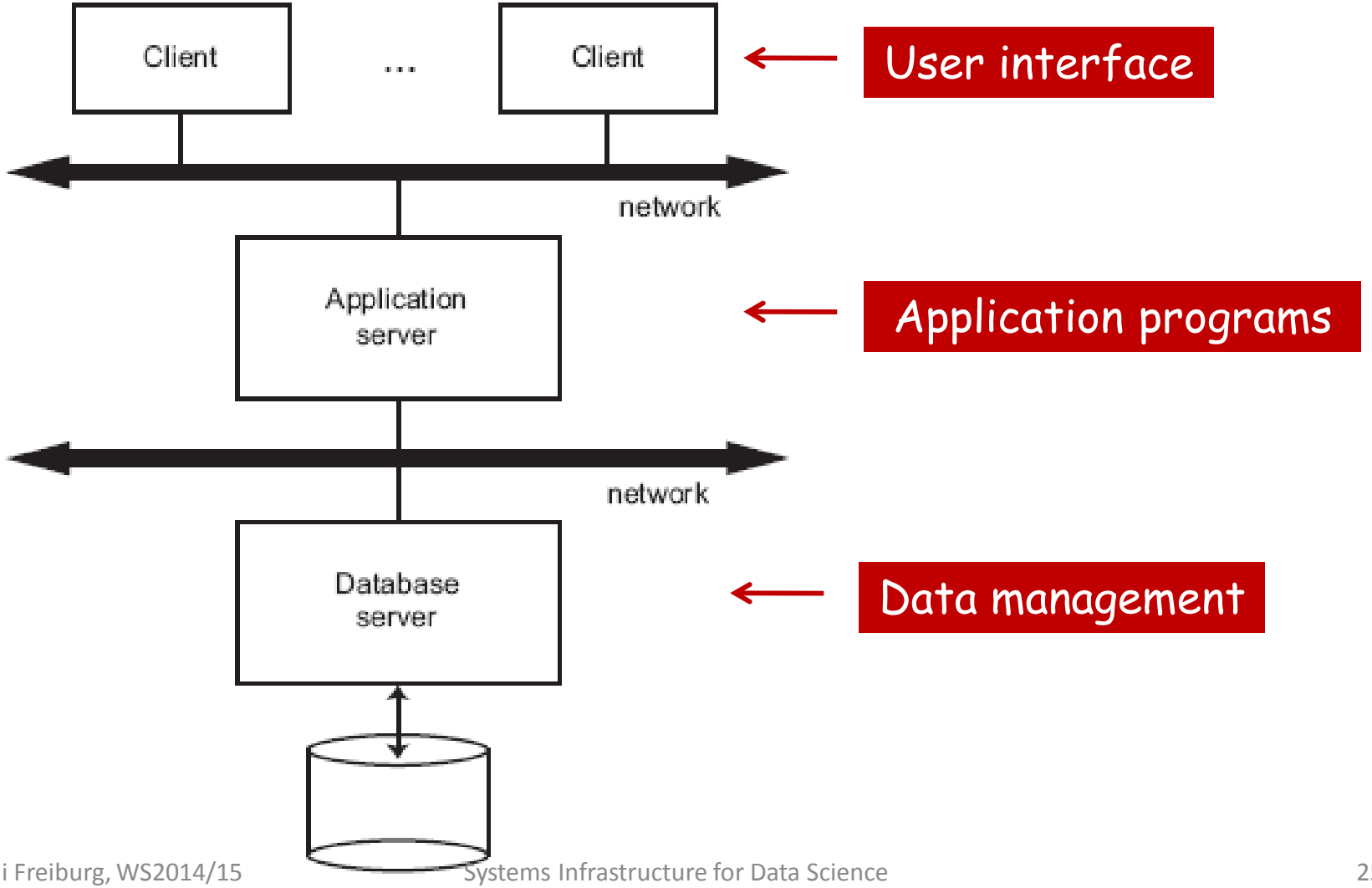
Important Architectural Dimensions for Distributed DBMSs



Client/Server DBMS Architecture



Three-tier Client/Server Architecture



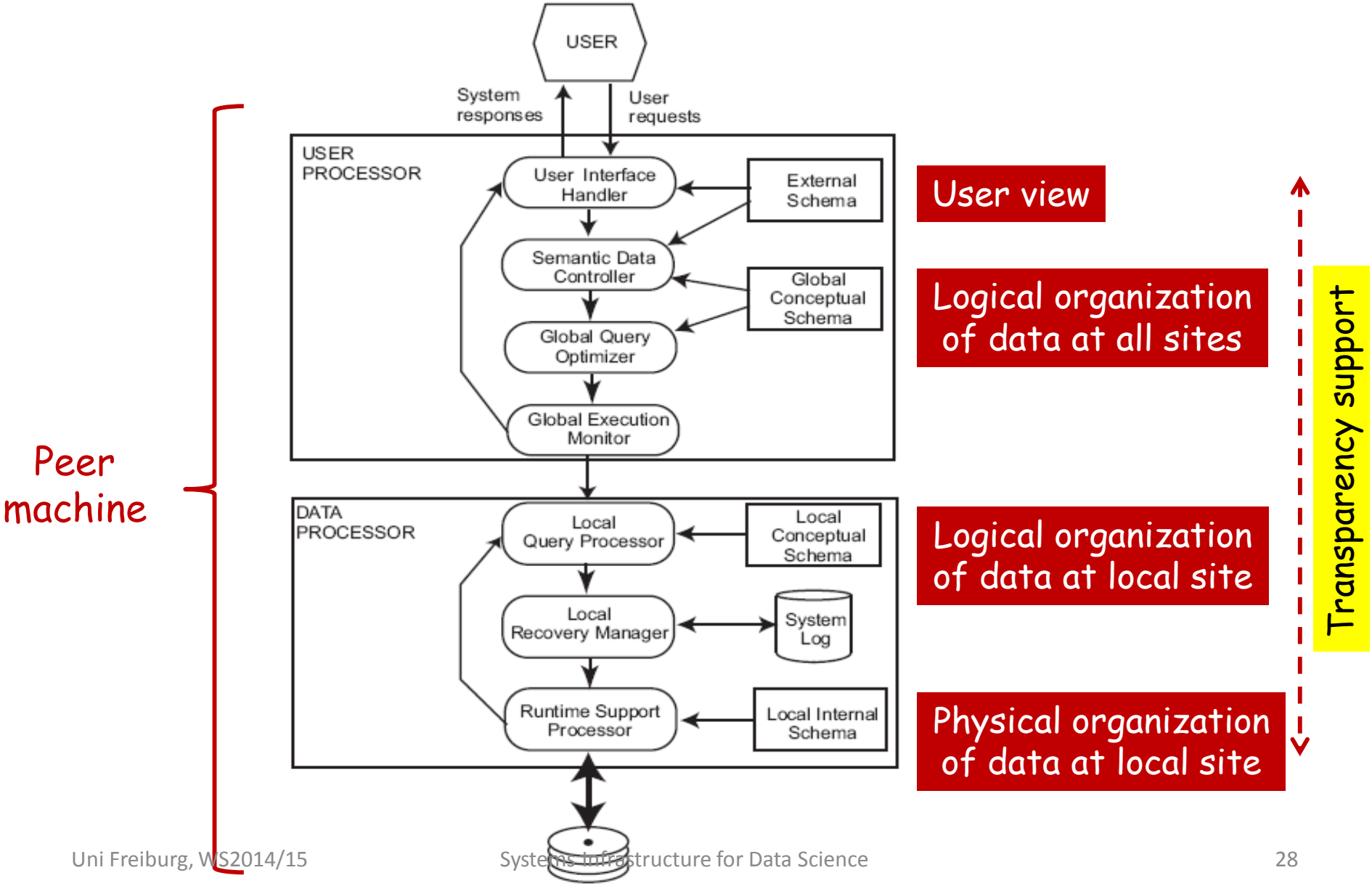
Extensions to Client/Server Architectures

- Multiple clients
- Multiple application servers
- Multiple database servers

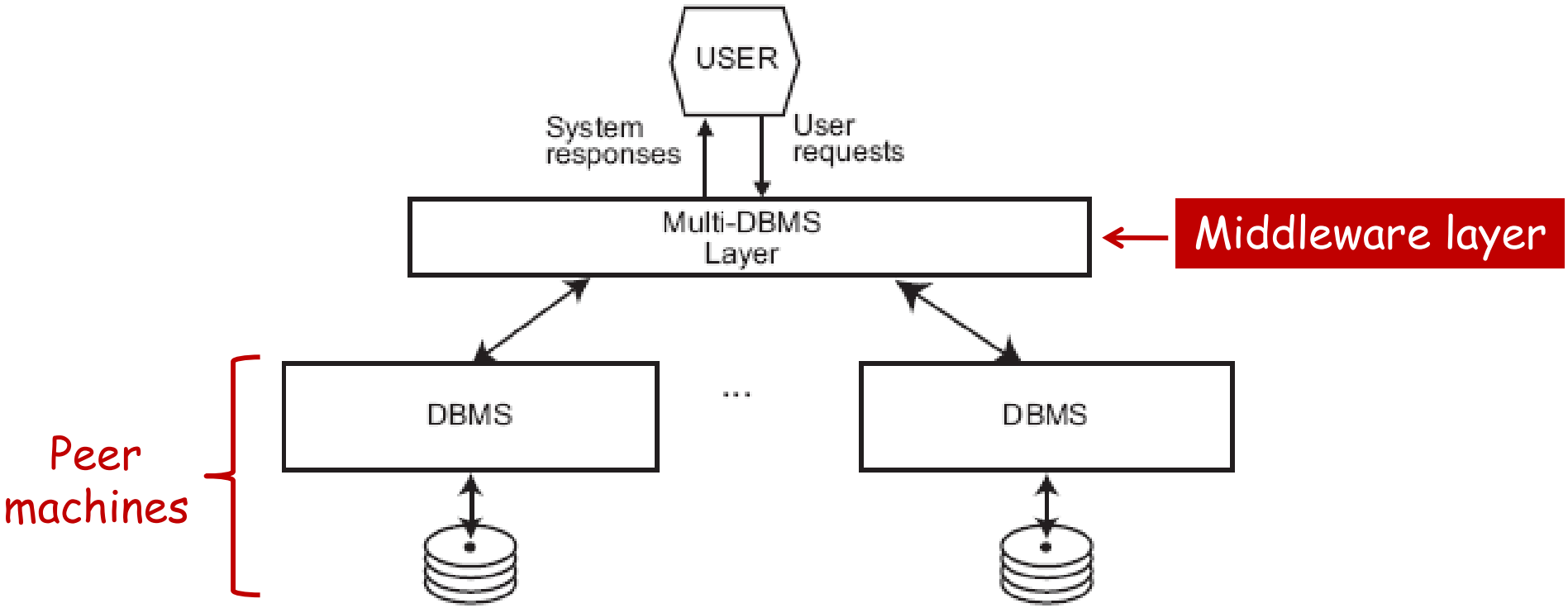
Peer-to-Peer DBMS Systems

- Classical (same functionality at each site)
- Modern (as in P2P data sharing systems)
  - Large scale
  - Massive distribution
  - High heterogeneity
  - High autonomy

Classical Peer-to-Peer DBMS Architecture



Multi-database System Architecture



- Full autonomy
- Potential heterogeneity

What is a Distributed DBMS?

- Distributed database:
  - *"a collection of multiple, logically interrelated databases distributed over a computer network"*
- Distributed DBMS:
  - *"the software system that permits the management of the distributed database and makes the distribution transparent to the users"*
- This definition is relaxed for modern networked information systems (e.g., web).