Lecture 1

Introduction to 5COSC004W

Client-Server

Architecture

Dr. Gabriele Pierantoni

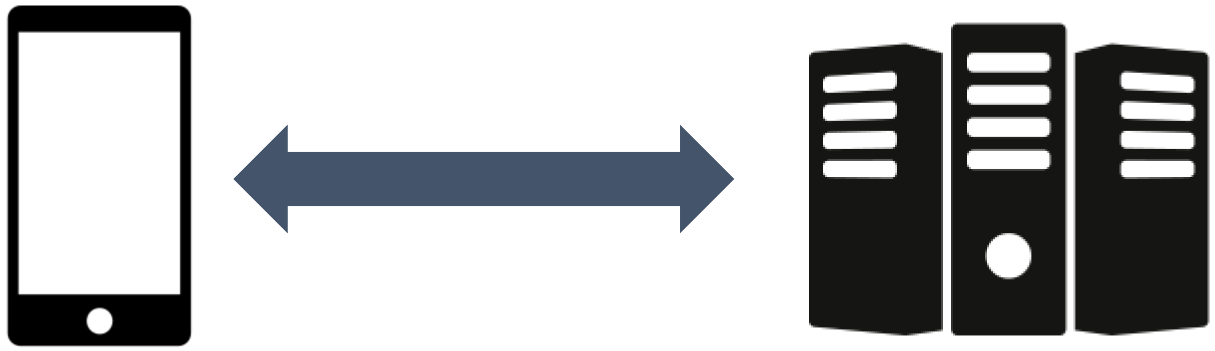
20.01.2021

Content of Today’s Lecture

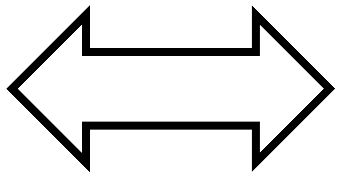
- Introduction to Client/Server Architecture

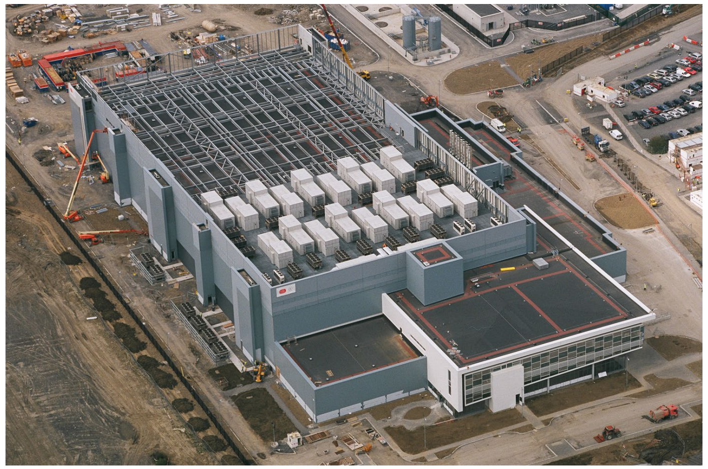
Local and Remote Programming

- Up to know, you have only studied programs that run in your local machine but most of the programs you use in your daily life (google, Facebook, email) does not run on your local phone/laptop.



Client-Server Paradigm



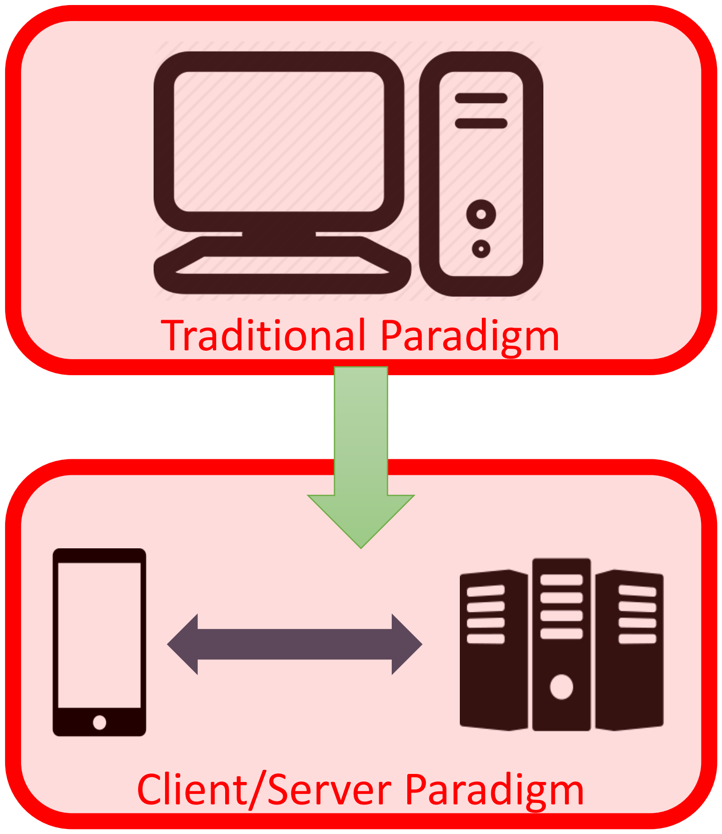


Client

Server(s)

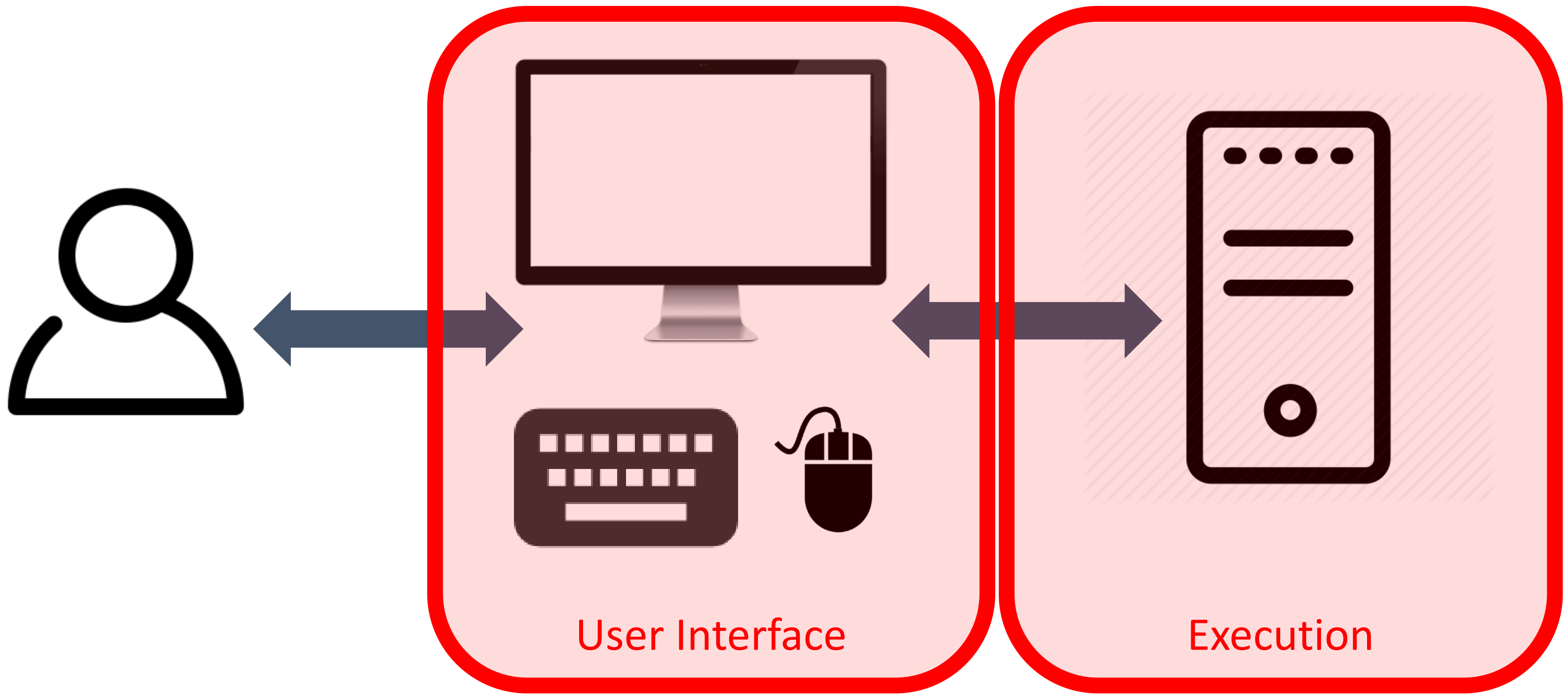
Client-Server Paradigm

Whenever you move a program (or part of a program) from the device it is accessed from (traditional architecture) to a client-server architecture, an entire new domain of programming opens up that covers many facets.



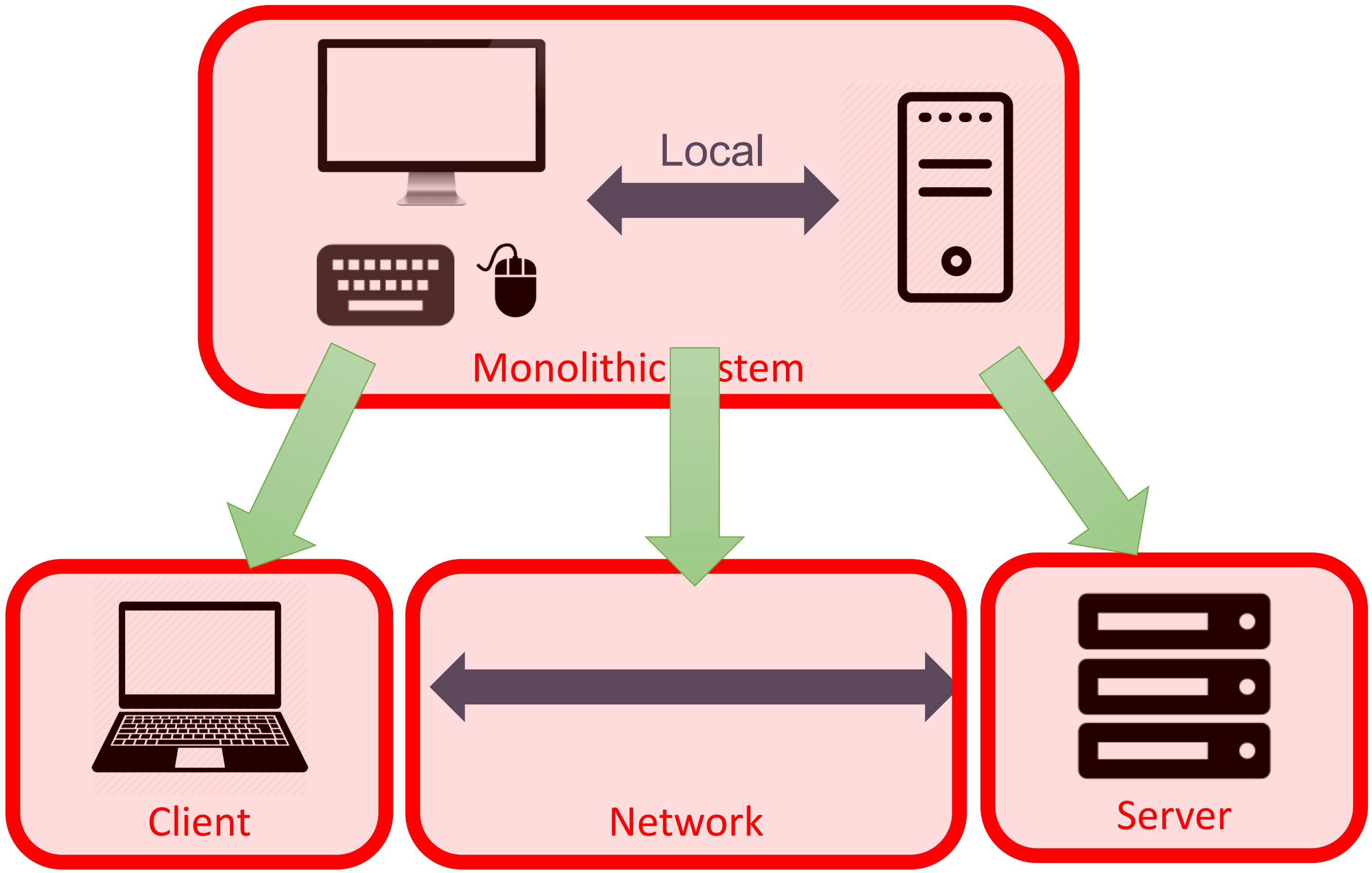
Traditional Programming

The interactions between the user and the system and the execution of the program all happen in the same system. That is easy !



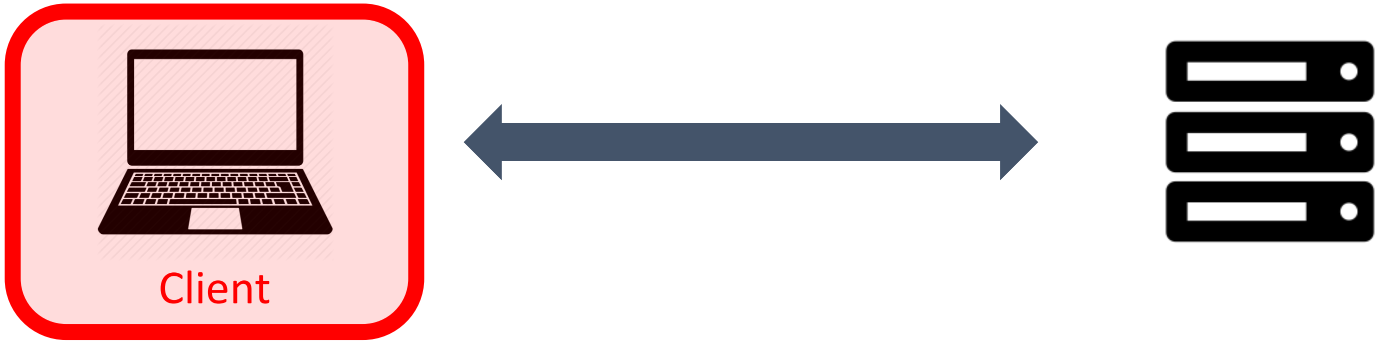
Client-Server Programming (1)

Let’s separate the user interface and the main execution of the program, i.e. two basic components of the system.



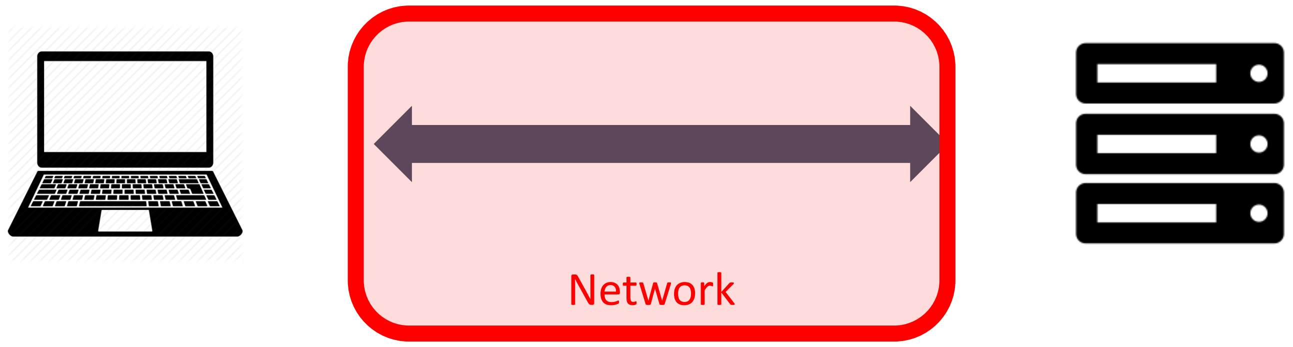
Client-Server Programming (2)

Clients: how to program the client-side.



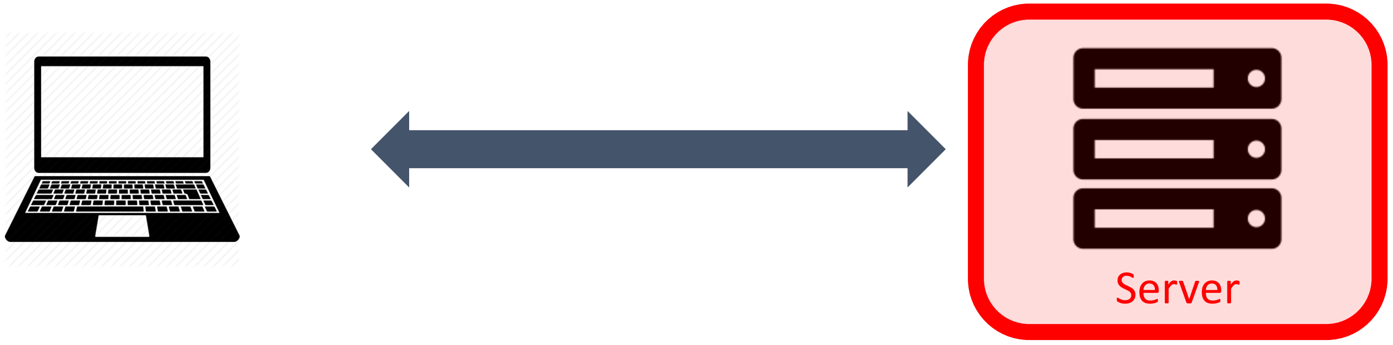
Client-Server Programming (3)

Network: how to program the network-side



Client-Server Programming (4)

Server: how to program the server-side.



Client-Server Paradigm

Client-Server Paradigm (1)

Definition (wiki)

- **Based on the distributed model having an architecture that partitions tasks or workloads between the providers of a resource or service (server) and service requesters (client) that communicate over computer network**
- **Server host runs one or more server programs which share their resources with clients**
- **Client does not share any of its resources, but requests a server's content or service function**
- **Server software accepts requests for a service from client software and returns the results to the client**
- **Client-server architecture separates a program from the device it is accessed from**

Client-Server Paradigm (2)

Components

client: active initiators of transactions making requests for services

- **user point of entry into a network**
- **normally a desktop computer, workstation, or laptop (or phone)**
- **user generally interacts directly only with the client portion software of an application**

example: web browser

server: manages hardware and software resources to process requests

- **servers are passive and react to client requests**

example: web server

communication network: connects clients and servers through computer networks

Client-Server Paradigm (3)

Server

- Starts first
- Passively waits for requests from clients
- Processes the request
- Responds to requests

Client

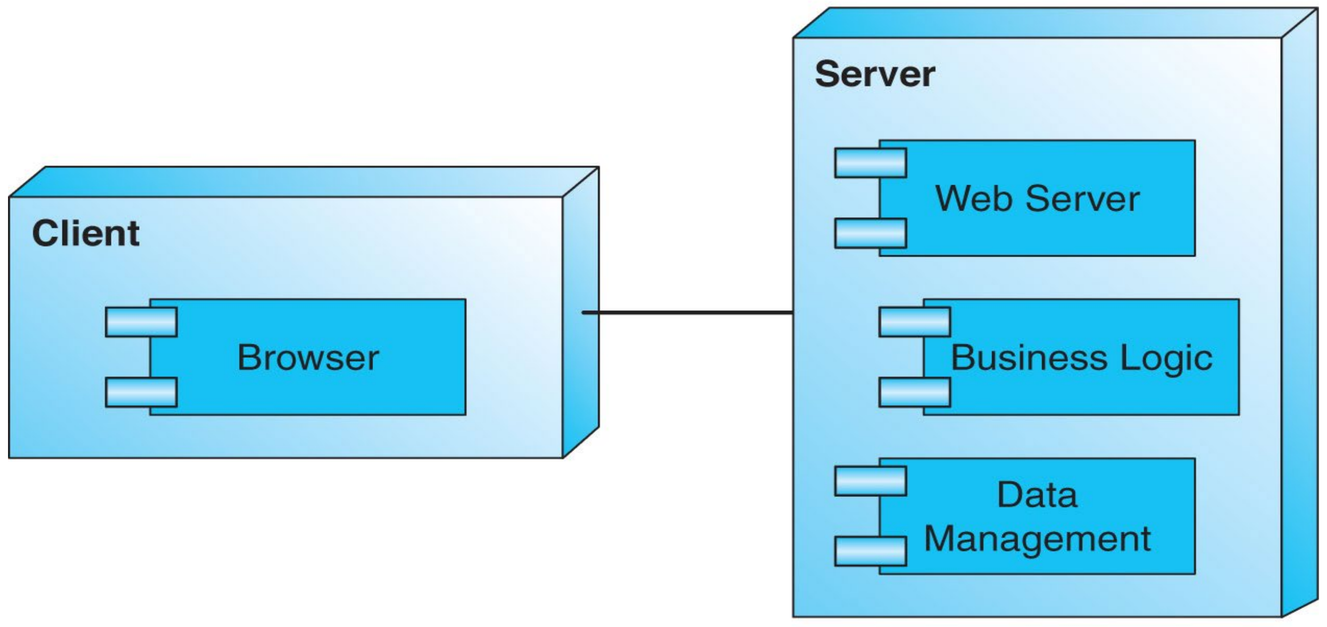
- Starts second
- Actively contacts a server with a request
- Waits for response from server
- Resumes the application

Client-Server Architecture

Client-Server Architecture: 2-Tier Architecture (1)

– Tier 1:
  - client platform, hosting a web browser

– Tier 2:
  - server platform, hosting all server software components



Tier 1:

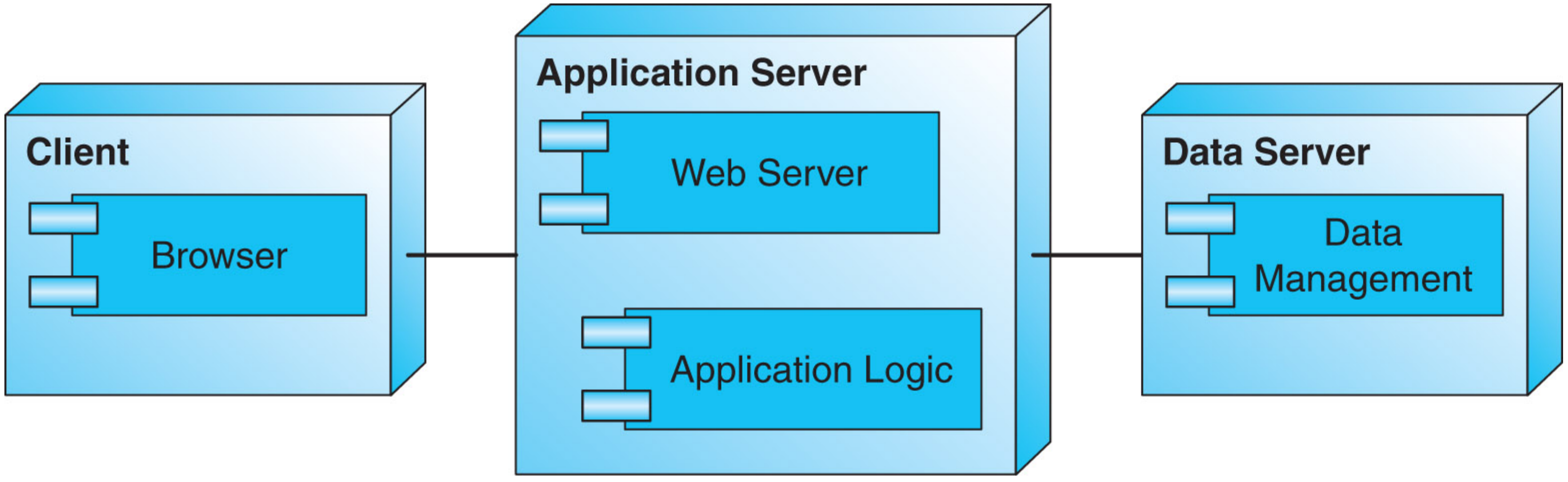
Tier 2:

Client-Server Architecture: 2-Tier Architecture (2)

- **Typical application**
  - **10-100 users**
  - **Small company or organization, e.g., law office, medical practice, local non-profit**
- **Advantage:**
  - **Inexpensive (single platform)**
- **Disadvantages**
  - **Interdependency (coupling) of components**
  - **No redundancy**
  - **Limited scalability**

Client-Server Architecture: 3-Tier Architecture (1)

– Tier 1:
  - client platform, hosting a web browser
– Tier 2:
  - application server platform - hosting application logic and web server
– Tier 3:
  - data server platform



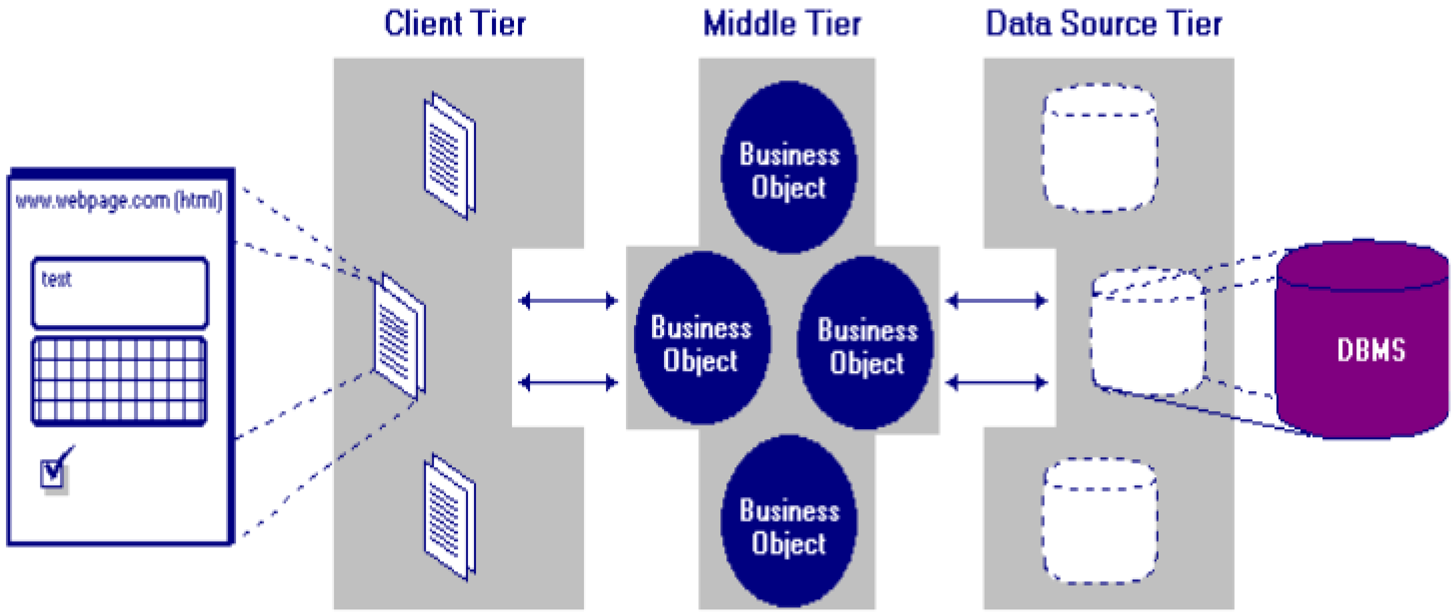
Tier 1: Tier 2: Tier 3:

Client-Server Architecture: 3-Tier Architecture (2)

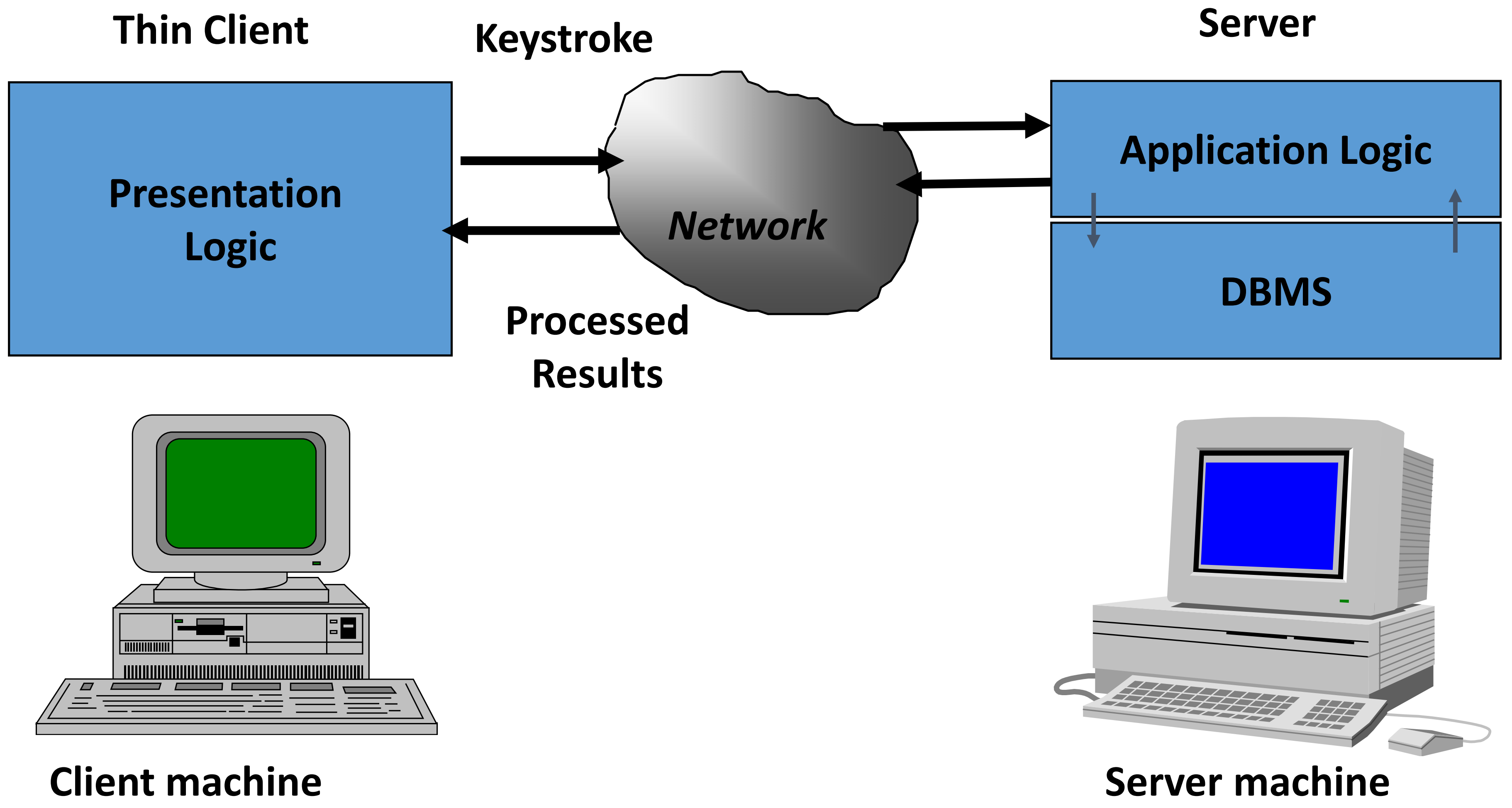
- **Typical Application**
  - **100-1000 users**
  - **Small business or regional organization, e.g., specialty retailer, small college**
- **Advantages**
  - **Improved performance, from specialized hardware**
  - **Decreased coupling of software components**
  - **Improved scalability**
- **Disadvantages**
  - **No redundancy**

Client-Server Architecture: Multi-Tier Architecture

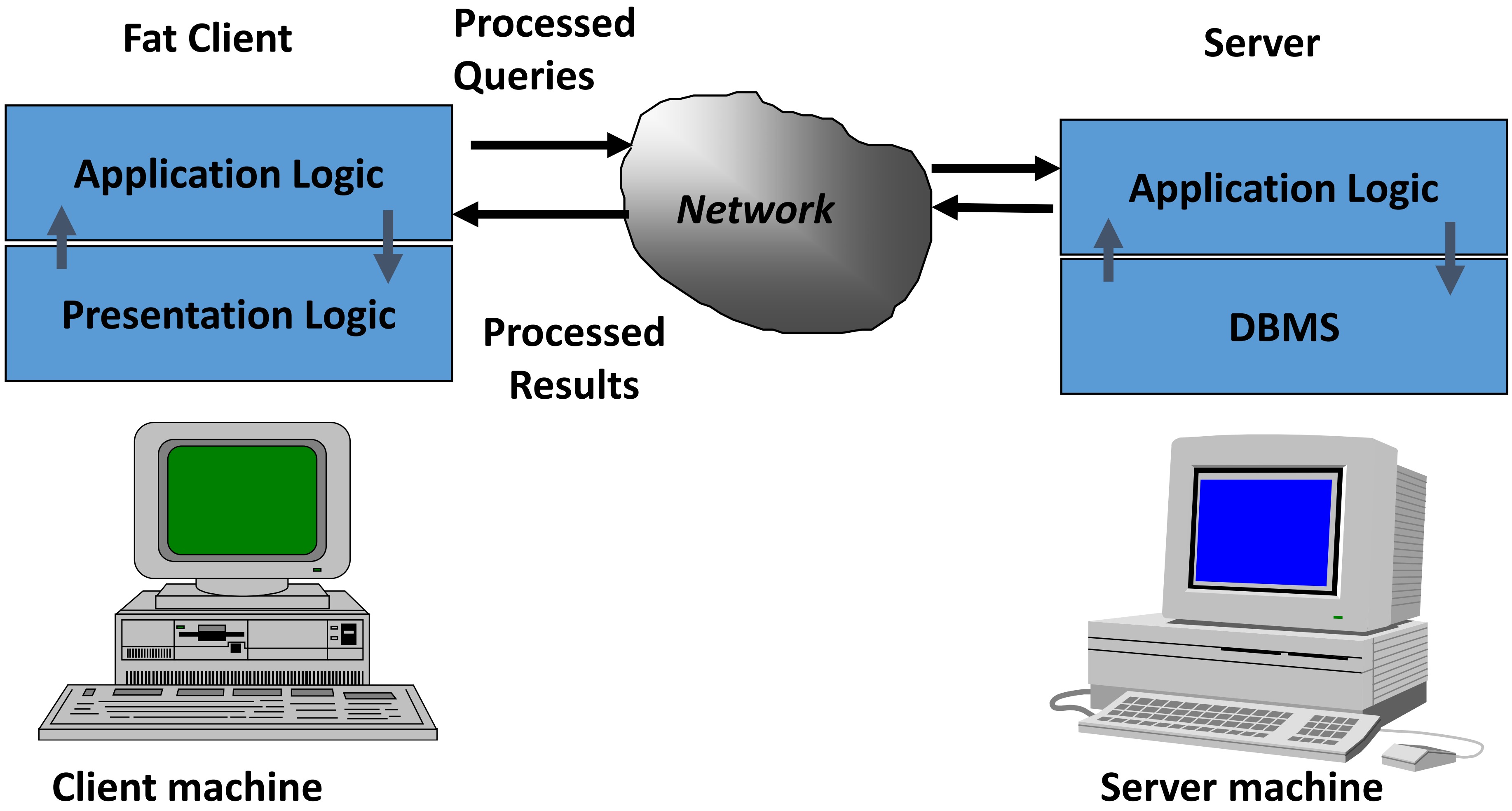
– **expansion of the 3-tier architecture, in one of several different possible ways:**
  - **replication of the function of a tier**
  - **specialization of function within a tier**



Client-Server Architecture: Basic Client-Server Model

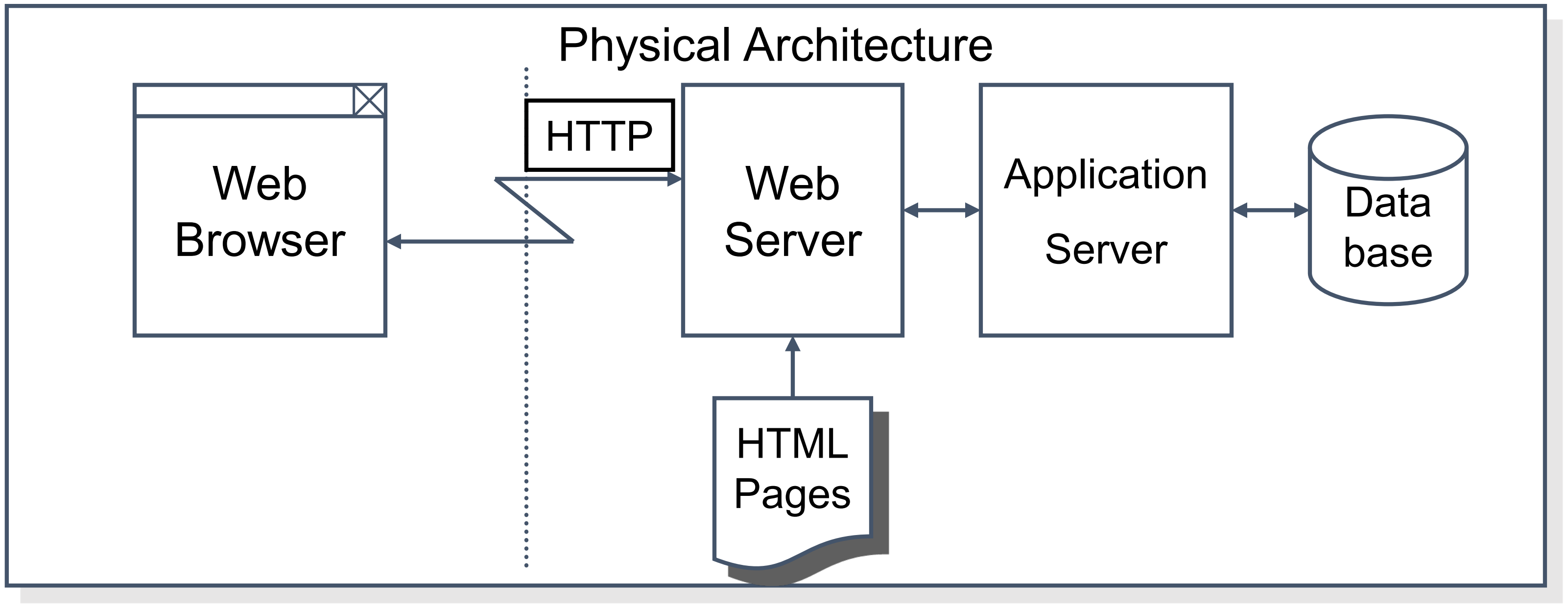


Client-Server Architecture: Distributed Client-Server Model



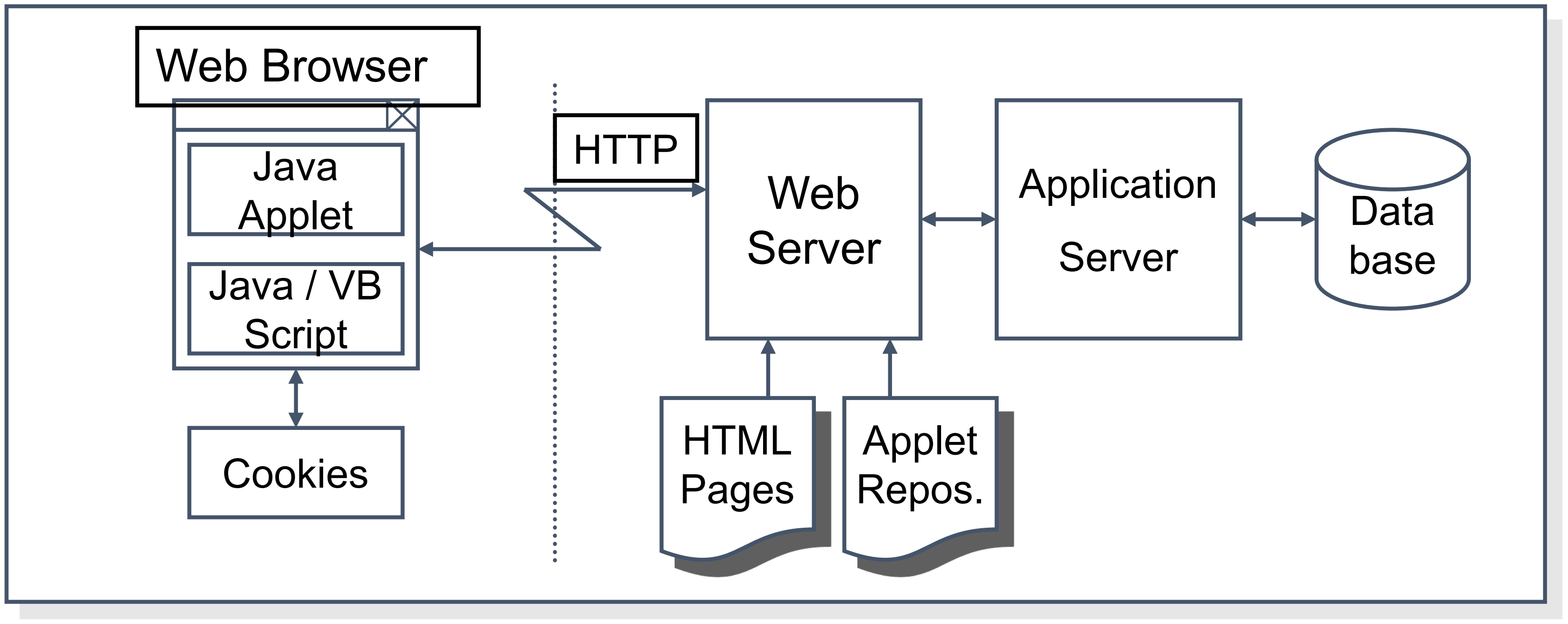
Client-Server Architecture: Thin Client Model

- **no extra software distribution required on the client computer**
- **it connects to server for every little action (e.g. input validation) but no immediate feedback on actions**
- **limited user interface design options with semi-static GUI**



Client-Server Architecture: Fat Client Model

- **automatic software distribution with software deployed on the client computer**
- **runs without connection**
- **active and nicer GUIs with immediate response**


Web Browser
Java Applet
Java / VB Script
Cookies
HTTP
Web Server
Application Server
Data base
HTML Pages
Applet Repos.