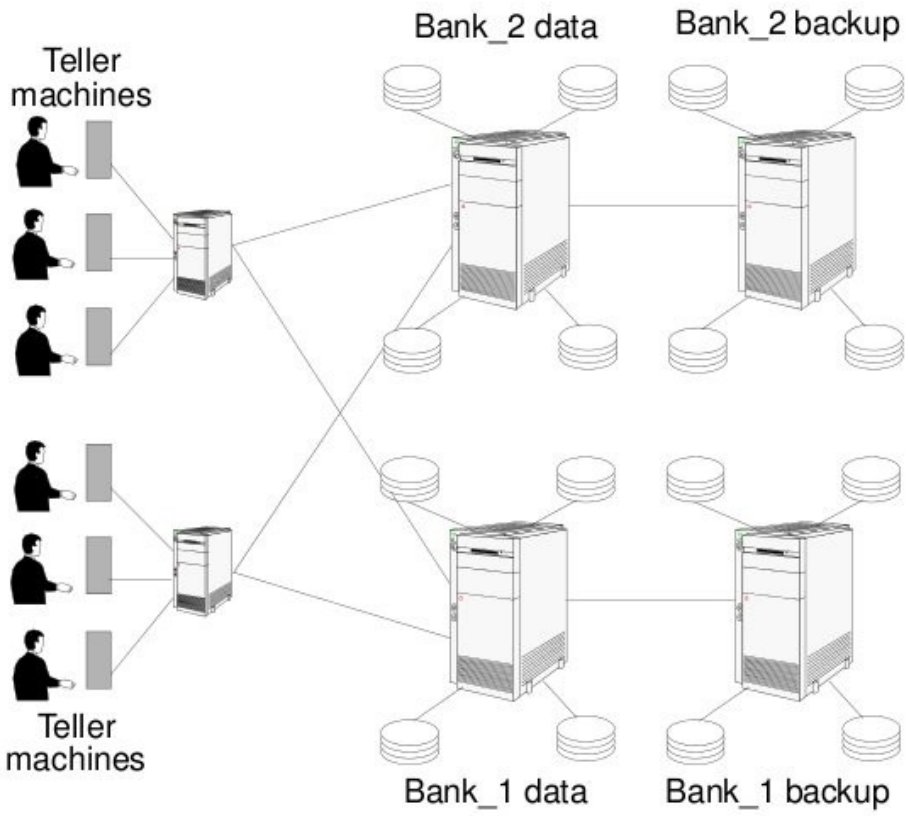
DISTRIBUTED SYSTEM - AN INTRODUCTION

A distributed system:
– Multiple connected CPUs working together
– A collection of independent computers that appears to its users as a single coherent system
• Examples: parallel machines, networked machines

A distributed system is the collection of independent computers that appears to the user of the system as a single computer. This definition has two aspects. The first one deals with the hardware: the machine are autonomous. The second one deals with the software, the users think of the system as a single computer

Example of distributed system: Automatic banking (teller machine) system



Advantages:

- **Performance:** very often a collection of processors can provide higher performance (and better price/performance ratio) than a centralized computer.
- **Distribution:** many applications involve, by their nature, spatially separated machines (banking, commercial, automotive system).
- **Reliability** (fault tolerance): if some of the machines crash, the system can survive.
- **Incremental growth:** as requirements on processing power grow, new machines can be added incrementally.
- **Sharing of data/resources**: shared data is essential to many applications (banking, computer-supported cooperative work, reservation systems); other resources can be also shared (e.g. expensive printers).

- **Communication:** facilitates human-to-human communication.

Disadvantages:

Difficulties of developing distributed software: how should operating systems, programming languages and applications look like?

Networking problems: several problems are created by the network infrastructure, which have to be dealt with: loss of messages, overloading, ...

Security problems: sharing generates the problem of data security.

Design issues that arise specifically from the distributed nature of the application

Transparency
Communication
Performance & scalability
Heterogeneity
Openness
Reliability & fault tolerance
Security

Transparency:

How to achieve the single system image?
How to "fool" everyone into thinking that the collection of machines is a "simple" computer?

Access transparency

- local and remote resources are accessed using identical operations.

Location transparency

- users cannot tell where hardware and software resources (CPUs, files, data bases) are located; the name of the resource shouldn't encode the location of the resource.

Migration (mobility) transparency

- resources should be free to move from one location to another without having their names changed.

Replication transparency

- the system is free to make additional copies of files and other resources (for purpose of performance and/or reliability), without the users noticing.

Example: several copies of a file; at a certain request that copy is accessed which is the closest to the client.

Concurrency transparency

- the users will not notice the existence of other users in the system (even if they access the same resources).

• Failure transparency

- applications should be able to complete their task despite failures occurring in certain components of the system.

• Performance transparency

- load variation should not lead to performance degradation.

This could be achieved by automatic reconfiguration as response to changes of the load; it is difficult to achieve.