Beyond the Client-Server Approach

Robin Sharp

Informatics and Mathematical Modelling
Technical University of Denmark
Phone: (+45) 4525 3749
e-mail: robin@imm.dtu.dk

Distributed systems

Definition: Systems, composed of a number of computers, which are connected by means of a communication network.

Def. covers a large number of possible practical organisations, including:

- Small systems, e.g. on a single chip
- Local systems, e.g. within a building or a department of a company.
- Large systems, with many computers spread over a large geographical area.

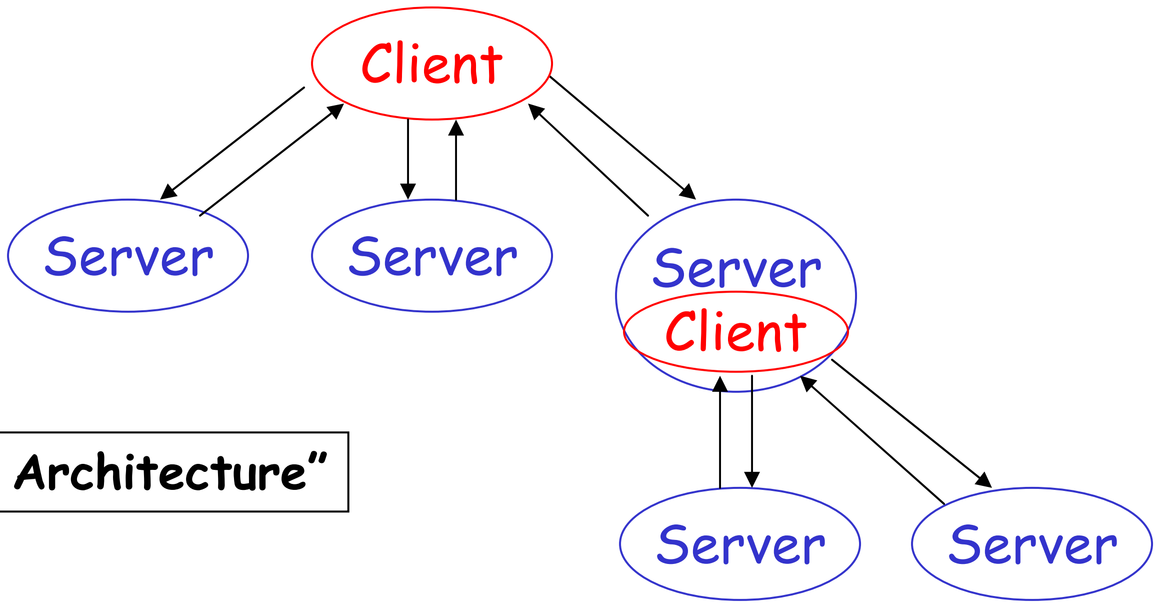
In distributed systems computations typically take place via collaboration between the computers.

Distributed applications

- Are based on communication via a (more or less reliable) Transport service — in the Internet, typically provided by TCP or UDP.
- Communication may support various ways of organising an application. Common examples:
  - Client/server: Two participants. One party (server) offers services to the other (client).
  - Peer-to-peer: Two or more participants with equal status.
  - Agent-based: Several parties collaborate in an “intelligent” way.
  - Grid: (Very) large number of parties offer services, and system will find the most appropriate one.

More Complex C/S Systems

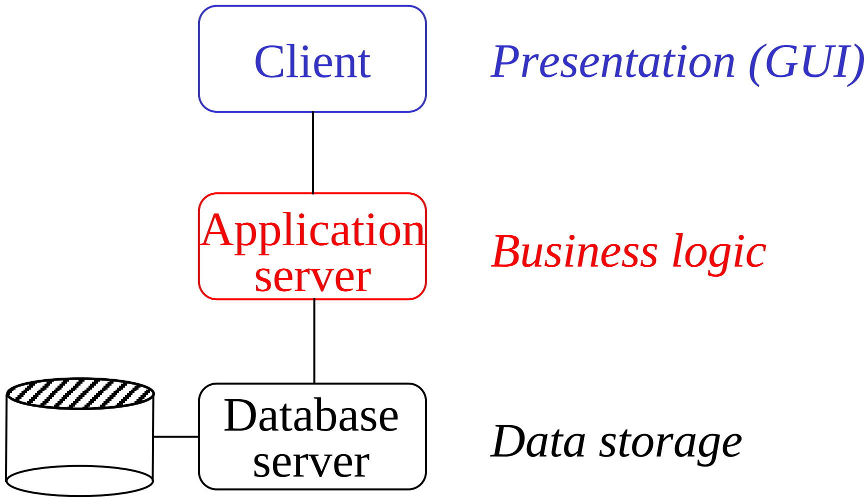
- In many applications, a simple “one server per client” architecture is not enough:
  - A client may need to get in touch with several servers.
  - A server may itself be a client of one or more other servers.



“Multi-tier Architecture”

Multi-tier Architecture

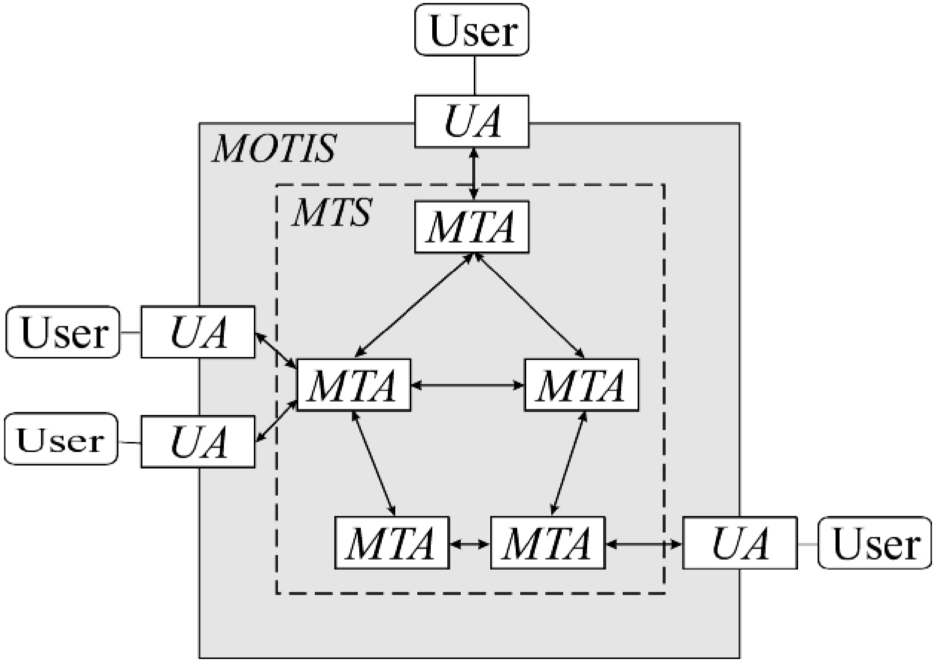
- Typical 3-tier example from a business application:



- Client looks after presentation and user input.
- Application server looks after specific "business logic"
- Database server looks after general data storage.

Agent-based systems

- Instead of a Client/server design, many interesting systems offer a service based on (possibly intelligent) collaboration between a set of ***agents***.
- Example: An agent-based Message Handling System.

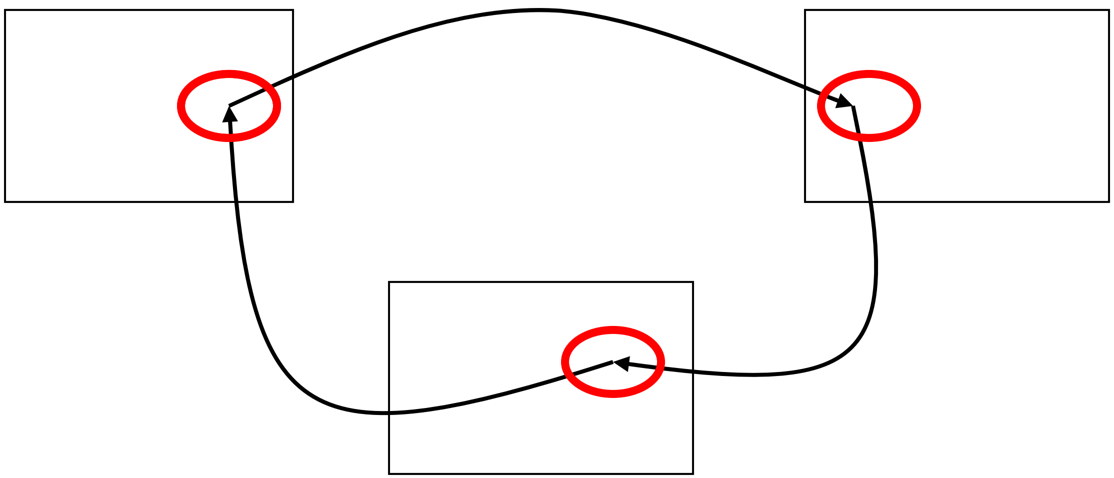


UA: User Agents, which handle interaction with users.

MTA: Message Transfer Agents, which pass messages.

Agent technologies

- Agents may be:
  - **Static:** An agent stays on a given system
  - **Mobile:** Agents move between systems collecting information.



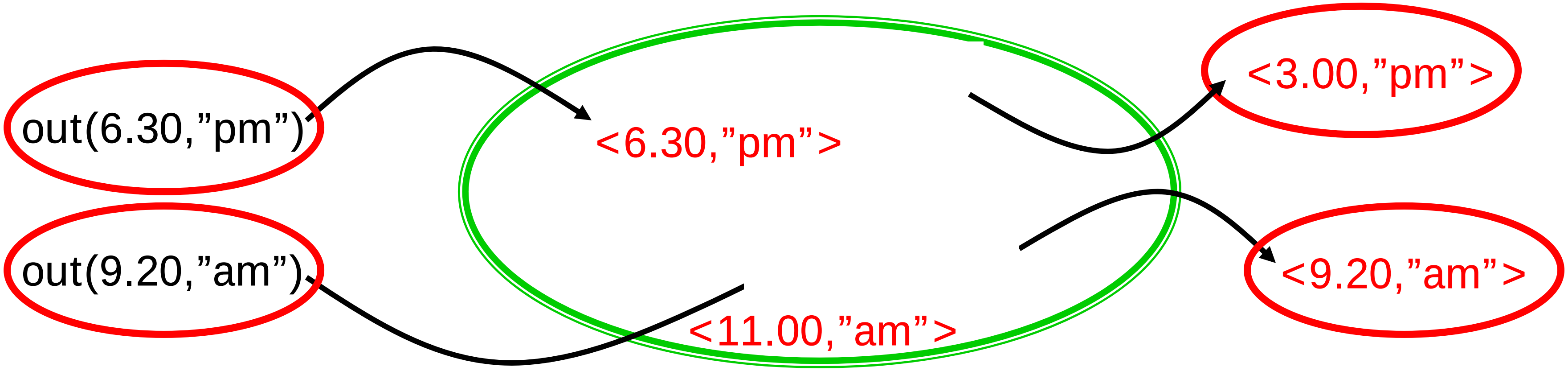
- **Security** is a big issue, especially for mobile agents. An agent should only have access to resources which it has permission to use!

Agent technologies (2)

- Agents may be designed on various abstraction levels:
  - **Coordination infrastructure**: System designer describes what the agents say to one another.

    Examples: JavaSpaces, Aglets, Jade, KQML.
  - **Coordination framework**: Uses a more abstract model of the system, such as **Shared Tuplespace**, to describe the actions of agents without saying explicitly how they communicate.

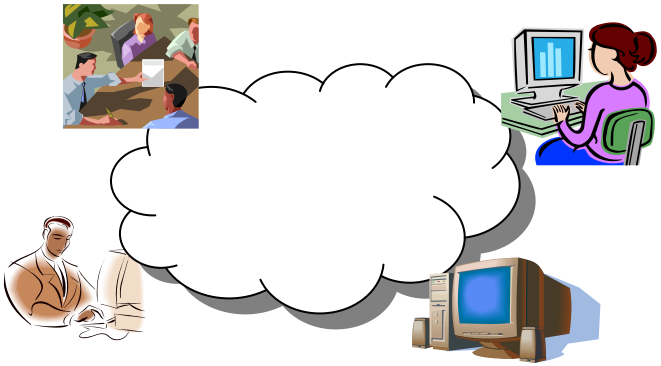
    Examples: Linda, TuCSoN.



Peer-to-peer (P2P) technology

- All participants have ***equal status*** and can communicate with each other without even knowing where their partners are.

  Examples: FreeNet, Gnutella, Chord.



- The practical implementation may involve one or more servers, which route communication and (in most cases) hide the clients from one another.

Grid Technology

- Recent proposals for very large distributed systems have focussed on distributing huge computations:
  - ***Molecular dynamics***
  - ***High energy physics***
  - ***Climate modelling***
  - ***Economic modelling***
  - ***Real-time global 3-dimensional illumination***
  - ***Extraction of information on Global Biodiversity***
  - ***Engineering simulations***
- Some such computations take several thousand CPU years or need storage for peta(10^{15})bytes of data.
- As with agents, the challenge is to distribute the activity for execution on a large number of systems, with adequate provision for security.

A simple idea: Aggregation

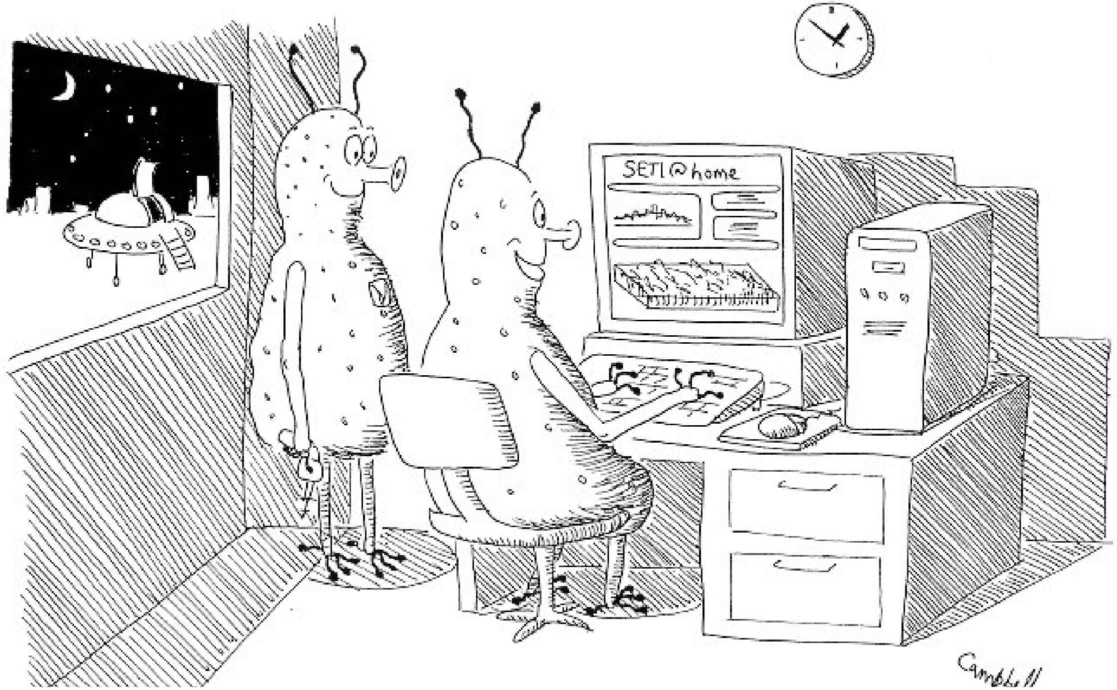
Exploit unused CPU time on millions of PCs to solve huge task sets.

Applications typically run as a client on each PC, which requests a new task from a server when it has spare CPU time available, e.g. as part of a screen saver.

- SETI: *Extra-terrestrial intelligence?*
  Analysis of billions of radio signal sequences.
- LifeSaver: *Cancer-active drugs?*
  Analysis of billions of potential active molecules.
- Quadratic/Number Field sieve: *Factors?*
  Analysis of billions of potential prime factors.

Projects such as these offer simple solutions which help mankind!

...probably

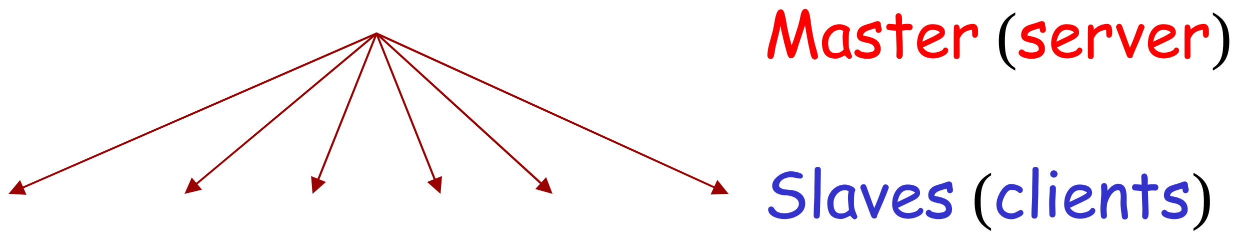


We’ll fiddle the output so there’s nothing unusual and these Earthlings will never find us!

Simple solutions

Simple solutions such as “screen-saver algorithms” work well when:

- Simple division into identical sub-tasks
- Simple dependency graph



- No special scheduling order (or deadlines).
- No communication between sub-tasks.
- Security same as for ordinary PC applications.

More difficult cases

- Architecture must allow us to access resources from a huge collection of heterogeneous computers in a uniform manner.
- This is the idea behind **Grid computing**:
  - **Computational grids**, which give access to storage and computational resources.
  - **Access grids**, which give access to information and presentation facilities, e.g. Teleconferences.
- Analogous to the power grid for supplying electricity!

The Grid

- In the **Power Grid**, users do not worry about where the electric power comes from – it just comes out of the electric plug!

- In a ***Computational Grid,*** users do not worry about where the computational power comes from!

Major challenges...

In a **Grid** with heterogeneous computers, covering a large physical area, there are challenges in (amongst other things):

- **Resource allocation:**
  - *How do we find the necessary resources?*
- **Scheduling:**
  - *In which order should the tasks be executed?*
- **Load balancing:**
  - *How do we ensure that the computers have "equal" loads?*
- **Communication:**
  - *How do we ensure that we use routes with lowest latency?*
- **Caching:**
  - *Must extra copies of data be stored nearby? How many?*
- **Security:**
  - *How do we secure the Grid system against hackers and other misuse?*

Further information...

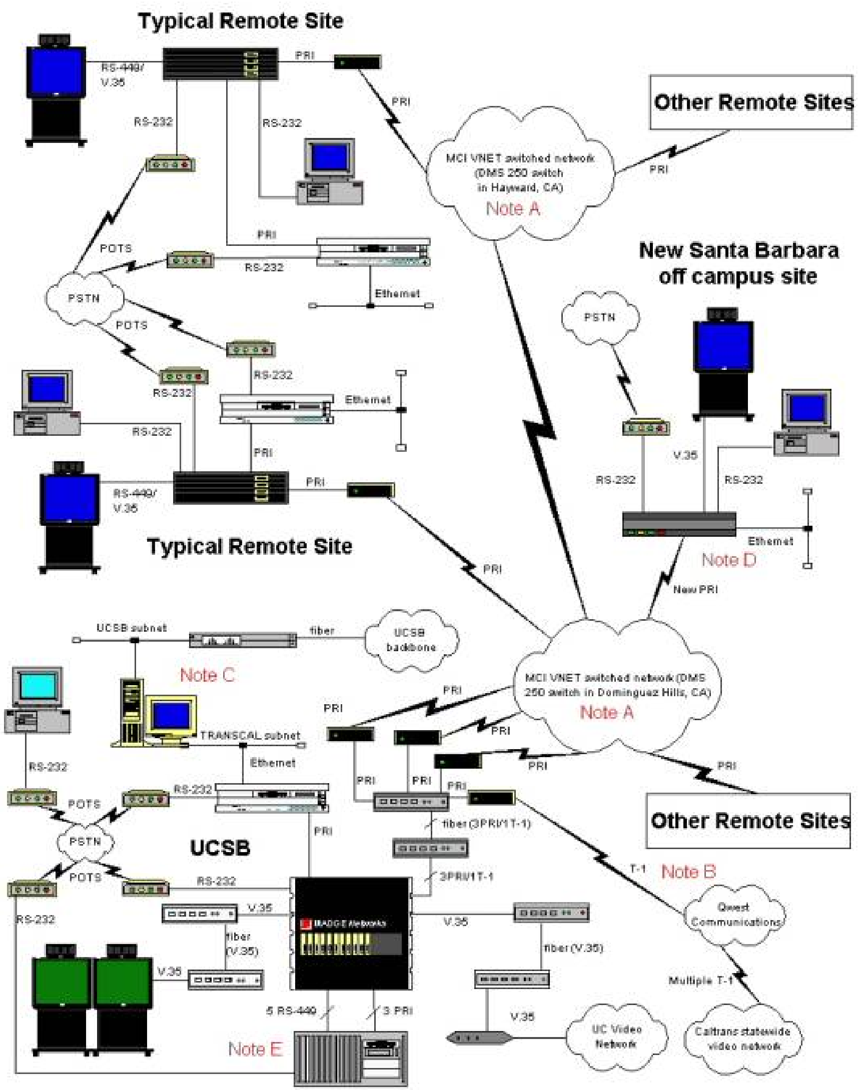
If you find these topics interesting, then consider following the courses:

02222: Distributed Systems (10 points, Spring)

- Paradigms, Protocols, Algorithms in systems of interconnected computers.

02345: Computer Security (10 points, Autumn)

- Applied cryptography, System design, Analysis of computer systems to ensure secure operation.



Thank you for your attention