Rozproszone systemy operacyjne

Jerzy Brzeziński

Jerzy.Brzezinski@put.poznan.pl

Poznań, 2003

Plan wykładu

1. Wstęp
2. Wymiana komunikatów
3. Komunikacja grupowa
4. Zdalne wywoływanie procedur
5. Rozproszona pamięć współdzielona
   - Konstrukcja mechanizmu dostępu
   - Modele spójności
   - Protokoły koherencji
6. Usługa nazw (naming)
   - NIS/NIS+
   - DNS
7. Sieciowy system plików
   - NFS, AFS, Coda
8. Synchronizacja
   - Synchronizacja zegarów
   - Wzajemne wykluczanie
   - Elekcja
9. Detekcja zakleszczenia

BIBLIOGRAPHY

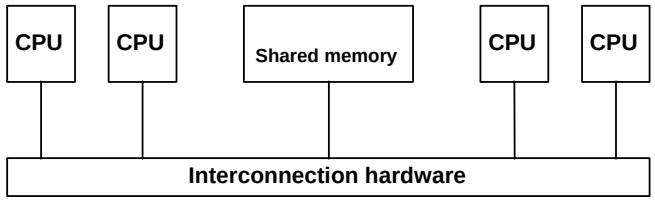
1. A. S. Tanenbaum, M. van Steen, *Distributed Systems: Principles and Paradigms*, Prentice-Hall, Inc., 2002.
2. P. K. Sinha, *Distributed Operating Systems - Concepts and Design*, IEEE Press, 1997.
3. A. S. Tanenbaum, *Computer Networks*, Pearson Education, Inc., 2003.
4. G. S. Hura, M. Singhal, *Data and Computer Communications. Networking and Internetworking*, CRC Press LLC, Boca Raton, Florida, 2001.
5. M. Singhal, N. G. Shivaratri, *Advanced Concepts in Operating Systems – Disitributed, Database, and Multiprocessor Operating Systems,* McGraw Hill, 1994.
6. A. Gościński, *Distributed Operating Systems, The Logical Design,* Addison Wesley, 1991.
7. A. Silberschatz, J. Peterson, P. Galvin, *Operating Systems Concepts,* Addison Wesley, 1991.
8. A. S. Tanenbaum, *Modern Operating Systems*, Prentice-Hall, Inc., 1992.
9. G. Tel, *Introduction to Distributed Algorithms*, Cambridge University Press, 1994.

Introduction

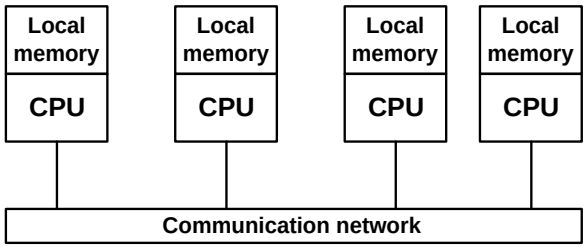
Basic Architectures of Multiprocessor Systems

Two basic types of computer architectures consisting of interconnected, multiple processors are distinguished: tightly coupled systems and loosely coupled systems.

Tightly coupled systems **-** systems with a single system wide primary memory (address space) that is shared by all the processors (also referred to as parallel processing systems, multiprocessors, SMMP - shared memory multiprocessors, SMS - shared memory systems, SMP – symmetric multiprocessors).



Loosely coupled systems - the systems where processors do not share memory and each processor has its own local memory (also referred to as distributed computing systems, multicomputers, DMS - distributed memory systems, MPP – massively parallel processors).



Distributed Computing System - Definition

Distributed computing system is a collection of independent computers (nodes, sites) interconnected by transmission channels, that appear to the users of the system as a single computer.

Each node of distributed computing system is equipped with a processor, a local memory, and interfaces. Communication between any pair of nodes is realized only by message passing as no common memory is available. Usually, distributed systems are asynchronous, i.e., they do not use a common clock and do not impose any bounds on relative processor speeds or message transfer times.

Evolution of Distributed Computing Systems

- **Early computers** :
  - very expensive (millions of dollars);
  - very large in size;
  - available only in research laboratories of universities and industries;
  - not accessible to ordinary users;
  - job setup time waste most of the valuable CPU time.

- **New concepts introduced to increase CPU utilization**:
  - *batching* similar job;
  - *automatic job sequencing;*
  - *off-line processing;*
  - *multiprogramming;*
  - *time sharing.*

- **Minicomputers:**
  - smaller, cheaper, having more processing capabilities than their predecessors;
  - computer resources shared simultaneously by many users;
  - computer accessed from a place different from the main computer room;
  - in the 1970s intelligent terminals replaced dumb terminals (combining the concepts of time sharing and off-line processing).

- **Workstations** :
  - available for only a small fraction of the price of minicomputers;
  - computing power almost equal to that of minicomputers;
  - used as terminals in time sharing systems in which most of the processing of a user‘s job could be done at the user‘s own computer, allowing the main computer to be simultaneously shared by a larger number of users.

Evolution of Distributed Computing Systems

Networking Technologies

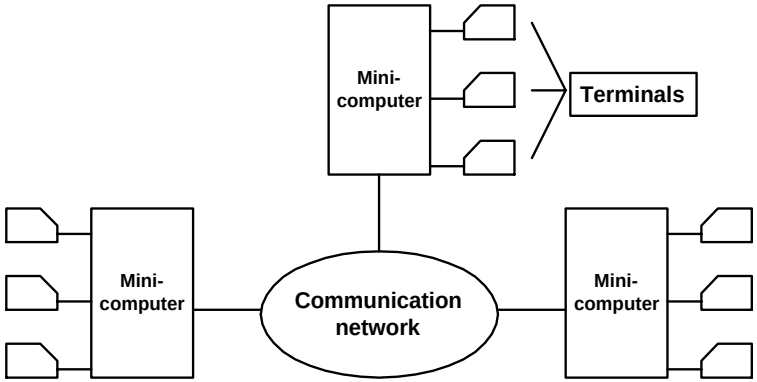
- **LAN - Local Area Network**:
  allows several computers located within a building or campus to be interconnected in such a way that these machines could exchange information with each other at data rates of about 10/100/1000/... Mbps;

- **WAN - Wide Area Network:**
  allows computers located far from each other (e.g. in different countries) to be interconnected in such a way that these machines could exchange information with each other at data rates of about 56Kbps/2/34/155/620/...Mbps;

The merging of computer and networking technologies gave birth to distributed computing systems in the late 1970s.

Distributed Computing System Models

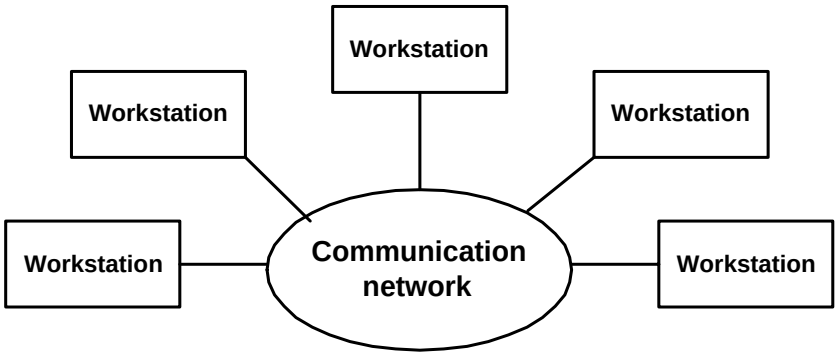
Minicomputer Model



Minicomputer Model – General Characteristic:

- Distributed computing system based on this model consists of a few minicomputers (or supercomputers) interconnected by a communication network.
- To each minicomputer are connected several interactive terminals.
- Each user is logged on to one specific minicomputer, with access to remote resources available on other minicomputers.
- Simple extension of the centralized time sharing system.
- The example of distributed computing system based on minicomputers is the early ARPAnet.

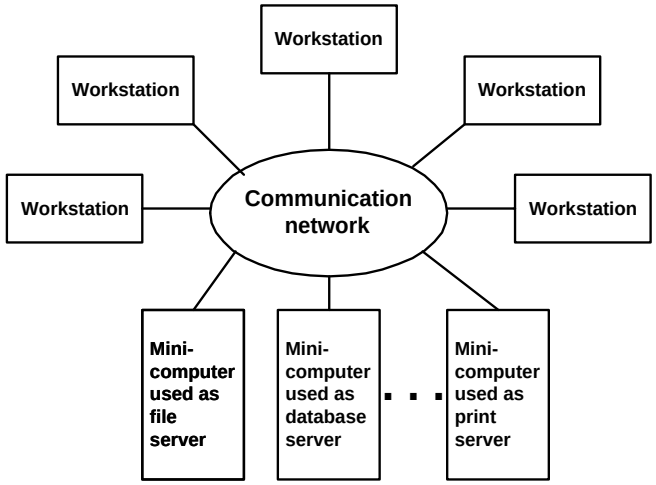
Workstation Model (NOW - Network of Workstations, P2P - Peer-to Peer)



Workstation Model – General Characteristic:

- System consists of several workstations interconnected by a communication network.
- Every workstation may be equipped with its own disk and serving as a single-user computer.
- In such environment like company‘s office or a university department, at any one time (especially at night), a significant portion of the workstation are idle, resulting in the waste of large amount of CPU time.
- Main idea: interconnect all workstations by a high-speed LAN so that idle workstations may be used to process jobs of users who are logged onto other workstations and do not have sufficient processing power at their own workstations to get their jobs processed efficiently.
- User logs onto one of the workstations and submits job for execution.
- If the user‘s workstation does not have sufficient processing power for executing the processes of the submitted job efficiently, it transfers one or more of the processes from the user‘s workstation to some other workstation that is currently idle and gets the process executed there.
- The result of execution is returned to the user‘s workstation.
- Implementation issues:
  - How does the system find an idle workstation?
  - How is the process transferred from one workstation to get it executed on another workstation?
  - What happens to a remote process if a user logs onto a workstation that was idle until now and was being executed a process of another workstation?
- Examples of distributed computing systems based on the workstation model:
  - Sprite system; experimental system developed at Xerox PARC.

Workstation - Server Model



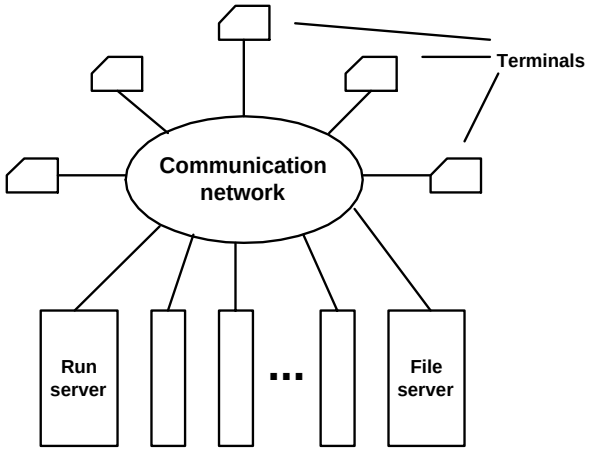
Workstation - Server Model – General Characteristic:

- System consists of a few minicomputers and several workstations (diskless or diskful) interconnected by a communication network.
- In addition to the workstation, there are specialized machines running server processes (*servers*) for managing and providing access to shared resources.
- Each minicomputer is used as a server machine to provide one or more types of service:
  - implementing the file system;
  - database service;
  - print service;
  - other types of service.
- User logs onto a workstation called his home workstation.
- Normal computation activities required by the user‘s processes are performed at the user‘s home workstation.
- Requests for services provided by special servers are sent to a server providing that type of service that performs the user‘s requested activity and returns of requested processing to the user‘s workstation.
- User‘s processes need not be migrated to the server machines for getting the work done by those machines.

Advantages of the Workstation-Server Model:

- It is much cheaper to use a few minicomputers equipped with large, fast disks that are accessed over the network than a large number of diskful workstations with each workstation having a small slow disk.
- Diskless workstations are also preferred to diskful workstations from a system maintenance point of view.
- Users have flexibility to use any workstation and access the files in the same manner irrespective of which workstation the user is currently logged on.
- This model does not need a process migration facility, which is difficult to implement.
- A user has guaranteed response time because workstations are not used for executing remote processes (the model does not utilize the capability of idle workstations).

Processor-Pool Model



Processor-Pool Model – General Characteristic:

- The model is based on the observation that most of the time a user does not need any computer power but once in a while he may need a very large amount of computing power for a short time.
- The processors are pooled together to be shared by the users as needed.
- The pool of processors consists of a large number of microcomputers and minicomputers attached to the network.
- Each processor in the pool has its own memory to load and run a system program or an application program.
- The processors in the pool have not terminals attached directly to them, and users access the system from terminals that are attached to the network via special devices.
- A special server (*run server*) manages and allocates the processors in the pool to different users on a demand basis.
- Appropriate number of processors are temporary assigned to user‘s job by the run server.
- When the computation is completed, the processors are returned to the pool.

Advantages of Processor-Pool Model:

- A user logs on to the system as a whole.
- It allows better utilization of available processing power.
- It provides greater flexibility than the workstation-server model.

Disadvantages:

- It is unsuitable for high-performance interactive applications.

Examples of Systems Based on Processor-Pool Model:

- Amoeba.
- Plan 9.
- Cambridge Distributed Computing System.

Hybrid Model – General Characteristic:

- Model is based on the workstation-server model but with addition of a pool of processors.
- The processors in the pool can be allocated dynamically for computations that are too large for workstations or that require several computers concurrently for efficient execution.
- The hybrid model:
  - gives guaranteed response to interactive jobs by allowing them to be processed on local workstation of the users;
  - is more expensive to implement than the workstation-server model or the processor-pool model.

Advantages of Distributed Systems over Centralized Systems

- **Scalability:**
  - capability of a distributed computing system to gradually and boundlessly in practice extend its power and functionality by simply adding additional resources (both hardware and software) when the need arises, without significant disruption of the normal functioning of the system (distributed computing systems that have the property of extensibility and incremental growth are called *open distributed systems*).

- **Efficiency:**
  - the total computing power of the multiple processors of distributed computing system can be utilized for providing shorter response times and higher throughput than a single centralized system.

- **Inherent Distribution:**
  - computing elements can be distributed to be available at appropriate locations for collecting, preprocessing, and accessing data as is required by many applications (e.g. computerized worldwide airline reservation systems, computerized banking systems in which a customer can deposit or withdraw money from his account from any branch of the bank, a factory automation systems controlling robots and machines all along an assembly line).

- **Economics:**
  - potentially a much better price/performance ratio than a single large centralized systems.

- **Reliability:**
  - *reliability* refers to the degree of tolerance against errors and component failures in a system; a reliable system prevents loss of information even in the event of component failures (the multiplicity of storage devices and processors allows the maintenance of multiple copies of critical information within the system and the execution of important computations redundantly to protect them against catastrophic failures).

Advantages of Distributed Systems over Independent Computers

- **Information Sharing:**
  - information generated by one of the users can be easily and efficiently shared by the users working at other nodes of the system (e.g. ***computer-supported cooperative working*** and *groupware*).

- **Resource Sharing and Utilization:**
  - sharing of software and hardware resources in a very effective way among all the computers and the users of a single distributed computing system.

- **Flexibility:**
  - the workload can be spread over the available computers in the most cost effective way.

- **Communication:**
  - human-to-human communication facilities are available and new communication services can easily be developed.

The advantages mentioned above are not achieved automatically but depend on the careful design of a distributed computing system.

Distributed Operating System

Operating system - a program that controls the resources of a computer and provides its users with an interface or virtual machine, that is more convenient to use than the bare machine.

The two primary tasks of an operating system:

- to present users with a virtual machine that is easier to program than the underlying hardware;
- to manage the various resources of the system.

Distributed operating system is one that looks to its users like an ordinary centralized operating system but runs on multiple, independent CPUs, i.e., the use of multiple processors is invisible (transparent) to the user (the user views the system as a „virtual uniprocessor", not as a collection of distinct machines).

Issues in Designing a Distributed Operating System

Transparency

- **Access Transparency:**
  - users should not need or be able to recognize whether a resource (software or hardware) is remote or local so the distributed operating system should allow users to access remote resources in the same way as local resources.

- **Location Transparency:**
  - name transparency - the name of a resource should be independent of the physical connectivity or topology of the system or the current location of the resource;
  - user mobility - no matter which machine a user is logged onto, he should be able to access a resource with the same name.

- **Replication Transparency**:
  - the existence of multiple copies of a replicated resource and the replication activity should be transparent to the users;

- **Failure Transparency:**
  - masking from the users‘ partial failures in the system, such as a communication link failure, a machine failure, or a storage device crash (system should continue to function, perhaps in degraded form, in the face of partial failures).

- **Migration Transparency:**
  - a movement of an object is handled automatically by the system in a user-transparent manner.

- **Concurrency Transparency:**
  - each user has a feeling that he is the sole user of the system and other users do not exists in the system.

- **Performance Transparency:**
  - the system can be automatically reconfigured to improve performance.

- **Scaling Transparency:**
  - the system can expand in scale without disrupting activities of the users.

Issues in Designing a Distributed Operating System

Reliability

- **Fault Avoidance:**
  - designing the components of the system in such a way that the possibility of fault occurrence is minimized.

- **Fault Tolerance:**
  - the ability of a system to continue functioning in the event of partial system failure; the performance of the system might be degraded due to partial failure, but otherwise the system functions properly.

- **Fault Detection and Recovery:**
  - the use of hardware and software mechanisms to determine the occurrence of a failure and then to correct the system to a state acceptable for continued operation.

The main drawback of increased system reliability is potential loss of time efficiency due to the extra overhead involved in described above techniques.

Issues in Designing a Distributed Operating System

Flexibility:

- ease of modification and enhancement.

Performance:

- the performance of distributed systems must be at least as good as a centralized system taking advantage of possible parallelism.

Scalability:

- the capability of a system to adapt to increased service load - a distributed system should be designed to easily cope with the growth of nodes and users in the system and such growth should not cause serious disruption of service or significant loss of performance to users.

Heterogeneity:

- different types of incompatibilities resulting from dissimilar hardware or software components of a heterogeneous distributed systems should not be visible to a user.

Security:

- protection of the various resources of a computer system against destruction and unauthorized access.

Emulation of Existing Operating Systems:

- a newly designed distributed operating system should be able to emulate existing popular operating systems so new software can be written using the system call interface of new operating system to take full advantage of its special features of distribution but vast amount of already existing old software can also be run on the same system without the need to rewrite it.

Models of an Operating System Kernel

- **Monolithic kernel model:**
  - most operating system services such as process management, memory management, device management, file management, name management and interprocess communication are provided by the kernel;
  - the large size of the kernel reduces the overall flexibility and configurability of the system but may improve efficiency.

- **Microkernel model:**
  - the kernel is a very small nucleus of software that provides only the minimal facilities necessary for implementing additional operating system services such as interprocess communication, low level device management and some memory management but all other operating system services are implemented as user-level server process;
  - easy to design, implement, install and maintain but possibly less efficient.