Distributed Operating Systems: One Big Machine and Amoeba

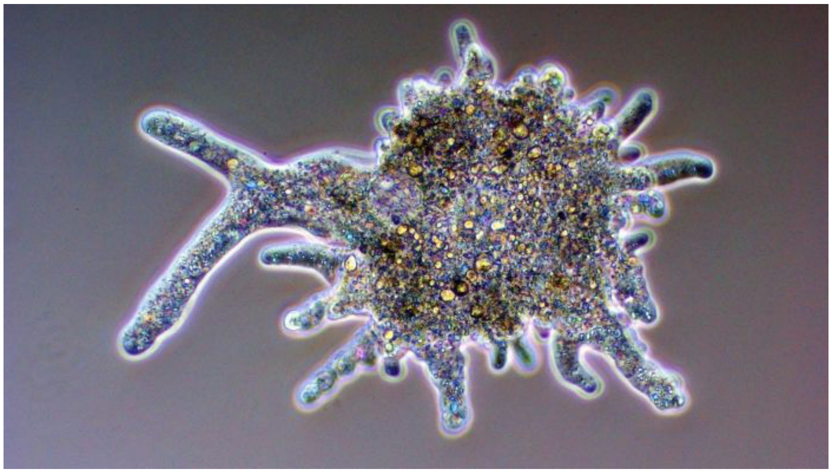
From computer cabinets to real Clouds

Priv.-Doz. Dr. Stefan Bosse
Universität Bremen, FB Mathematik & Informatik
19.3.2018
sbosse@uni-bremen.de

1. Inhalt

2. Introduction

Question: How can complex systems be implemented with simple concepts?



2.1. Goals

- You understand basic principles of a Distributed Operating System and the distinction from Networked Operating Systems
- You understand the basic challenges and features of distributed computing and distributed file systems
- You will be able to understand how Clouds of the future can be designed and used
- Some practical programming lessons should demonstrate simple network communication and distribution of computation

2.2. Distributed vs. Parallel Systems

Distributed System

A Distributed system is a collection of **loosely coupled** processors or computers interconnected by a communication network (**Multicomputers**)

- **Memory Model**: Distributed Memory → Each processor has private memory
- **Communication**: Message based using Networks

- **Resources**: Not directly shared

Parallel System

A Parallel system is a collection of **strongly coupled** processors (**Multiprocessors**)

- **Memory Model**: Shared Memory
- **Communication**: Directly via electrical signals → Switched Network (Crossbar) | Bus → Point-to-Point | Point-to-N Networks
- **Resources**: Shared (Bus, Memory, IO)

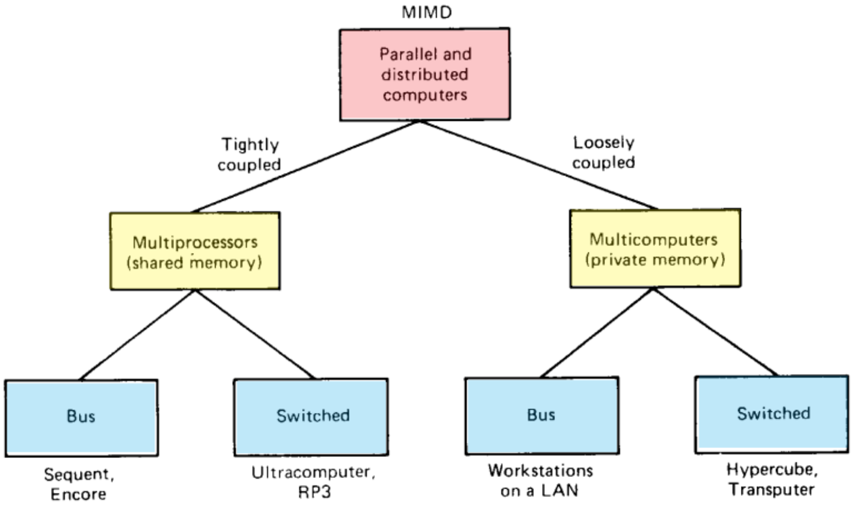
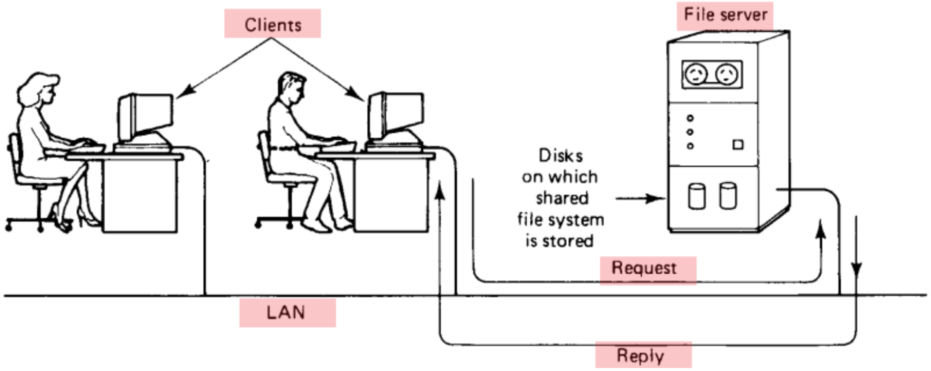


Fig. 1. Taxonomy of distributed and parallel systems [1]

2.3. Distributed Operating Systems

- Historically based on **Network Operating Systems** (e.g. Linux Clusters): *Users are aware of multiplicity of machines!*
  - Tools: Remote Login (`telnet`, `ssh`), Remote Desktop (Windows), File Transfer (`FTP`, `SSH`), Network Filesystem (`NFS`)
- A **Distributed Operating System** hides machines: *Users are not aware of multiplicity of machines!*
  - Access of remote resources similar to access of local resources
  - Transfer of computation (rather than data)



2.4. History

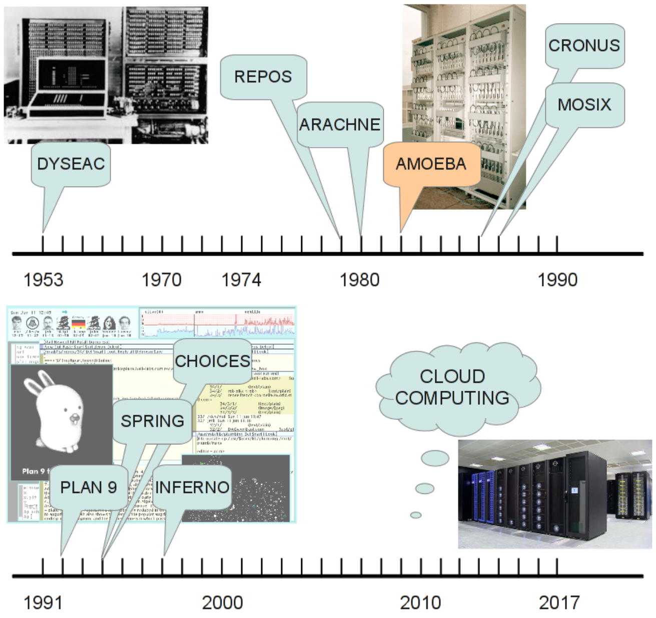


Fig. 2. Time line of some selected DOS. Golden age of DOS development was around 80's and 90's!

2.5. Objects and Filesystems

Resource == Object = {file, device, processor, memory, ..}

Traditional Operating Systems

- Filesystems handle mostly the organization and structuring of data
- UNIX: Local devices and processes are represented by virtual files

Distributed Operating Systems

- Handle all objects in the filesystem
- Provide name spaces and name mapping service: *resource* ⇔ *name*
- Plan9: All objects are files!
- Amoeba: File storage (data) and Organization (directories) separated!
  Objects are handled by servers → Object-orientated approach!

2.6. Distributed Operating Systems

Design and feature criteria

Naming
How can we name an object that is far away with unknown location?

Robustness
What happens if a machine or a network fails?

Security
How can we protect our system from failure, fraud, intrusion , hijacking, .. ?

Performance
Slower than ever?

Consistency
I made a bank transaction, the acknowledge of the transaction getting lost, and the transaction was repeated → my account was charged twice times?

Scalability
What happens with these criteria if we increase the number of machines by 10 times?

3. Distributed OS Amoeba

> *Question: What makes the Amoeba OS unique and different from UNIX or Windows?*

3.1. Goals

One Big Machine

- **Presentation**: Present a network of computers as a single machine to the user and programs
- **Transparency**: No difference in accessing local and remote resources (files, devices, processor, memory)
- **Computer Architecture**: Use of generic computers already available!

Resources

- All resources are handled as objects by servers with unified object descriptors: **Capabilities**

Future not Past

- Do not rely on any existing OS or concept to design a new efficient and clean DOS

Flexibility

- Easy to extend; scalable → The natural Amoeba!

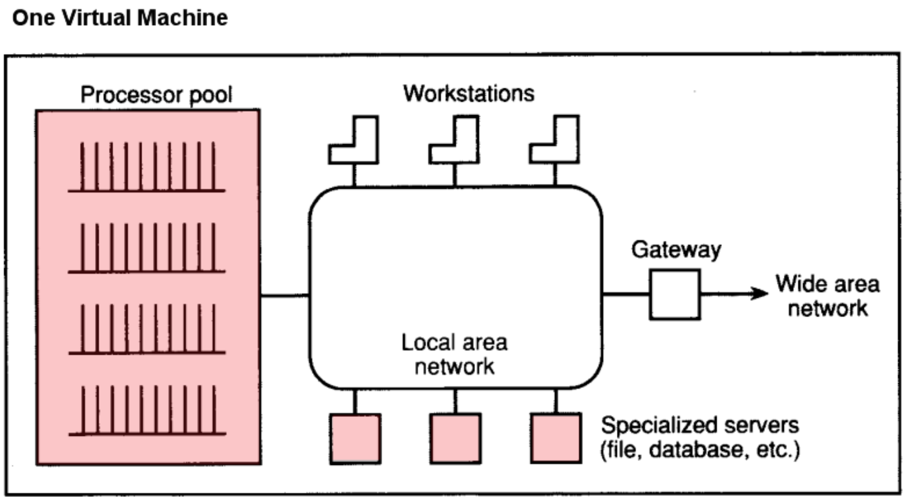
3.2. Hardware Architecture

Amoeba hardware consists of four components:

- Workstations
- Pool processors
- Storage

- Networks / Gateways

But all computers and components can be generic: Big server, desktop computer, mobile device (smartphone!), Embedded Computer (Raspberry PI!)



3.3. Software Architecture

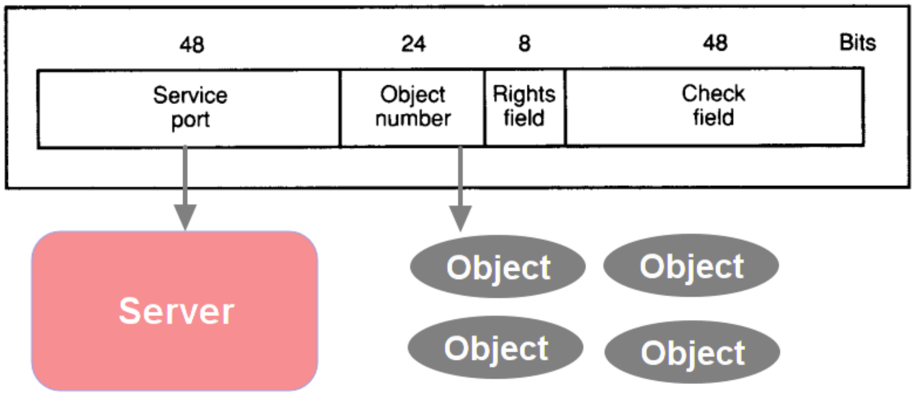
> *Amoeba is an object-orientated system using clients and servers. But the roles client/server are not fixed. A server can be a client, too!*

- Client processes (i.e., any program) use the concept of *Remote Procedure Calls* to send requests to servers and to get replies:

  RPC : *request* → *message* → **Server** → *message* → *reply*

- UNIX: Objects (files) are handled (identified) with paths
- Amoeba: All objects (files, processors, processes, devices, ..) are specified by and handled with single **capabilities**!
- A set of objects are handled by servers (e.g. file server, processor server, ..)
- Servers are processes that can be executed on *any* machine!

3.4. Object Capabilities

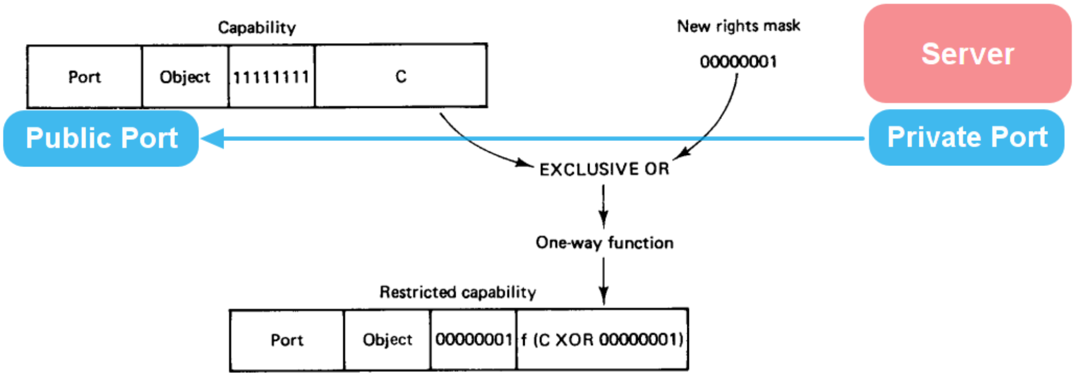


- A capability (object handle) is a record that contains the following data:
  - A server port number (e.g., 100239) linking the object (e.g. file) to a server (file server)
  - An object number identifying the object uniquely on *this* server
  - A rights field: Which operations are allowed with *this* capability? (read, write, execute, delete, ..)
  - A check field securing and protecting the capability and the rights field

3.5. Security

An object (e.g. file) can be represented by different capabilities allowing only a sub-set of operations (read, write, execute, delete) → restricted capabilities

- The check field encrypts the entire capability using a one-way function and contains the rights field
- Only servers can restrict capabilities: They use a private key to create a public key (check field)
- Server “listen” on private port, but clients access servers with encrypted public port → No server faking possible!



3.6. Servers and Services

- A server provides a service to access objects of the server → object-orientated approach
- A server manages a table containing objects. The object number is the table row. The columns are data.

Amoeba servers

BULLET Fileserver
The file server. This server only stores file data as linear blocks. A committed file is atomic and cannot be modified → Robustness!

SOAP Directoryserver
The name space server. It provides a directory graph with tables mapping names on capabilities. The directories are stored as Bullet files. More than one file server can be used in replication mode.

RUN Excecutionserver
The process server. It controls the execution of programs and supports process snapshot migration from one to another machine.

3.7. Communication

Long story - short conclusion

Distributed Systems require communication by using messages.

- Network communication is slow compared with memory access
- Major goal of DOS: Speedup by Parallelization!!
- But communication reduces the degree of Parallelization!

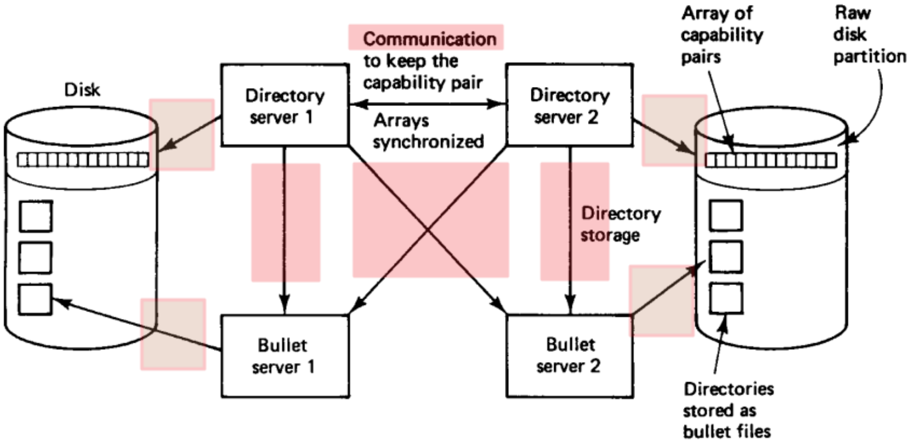


Fig. 3. Interaction between two SOAP directory server operating in two-copy mode and two BULLET file servers.

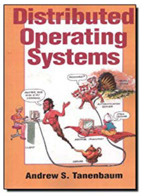
4. Summary

- The Amoeba OS poses a very simple and clean design princpile to compose large-scale distributed systems
- It is an object-orientated OS with servers managing objects
- Objects (files, screens, processes, ..) are accessed by using capabilities and Remote Procedure Calls
- Servers are identified by their ports - not by their location!
- Robustness: File server stores files in contiguous blocks. After a file is committed it is immutable!
- A directory-based naming service provides name → capability mapping
- Robustness: File and directory servers can be redundant

5. References and Further Reading

Books - Further Reading

1. Andrew Tanenbaum, Distributed Operatings Systems, Pearson, 1996. → Chapter 7



6. Practical Lessons

Using JavaScript and node.js

Group work!

1. You will get a JavaScript code template that provide basic operations/primitives:

    - Networking: Sending/receiving of text messages (RPC)
    - Synchronisation
    - Capabilities
    - Service Loop
    - File and Name server (sim. Bullet/SOAP)
    - ..

    You will be able to construct and study a very simple DOS based on the Amoeba principles using IP networks / the Internet. Use the provided demo application and set-up a small distributed network in your group and start the hello world service. Question: What is an IP address? How was it assigned to your computer? How do you find your IP address? UNIX/Windows?

2. Create a schematic diagram of your current network and explain the services that are provided. Read reference [1]/Ch.7 for more information.

    If your program communicates with servers (e.g., the hello world server) on a remote computer it has to have the IP address of the remote machine.

But you uses a capability? How can the mapping Server Port ↔ IP be resolved? How is it done in the original Amoeba OS?