Architectures

of

DISTRIBUTED SYSTEMS

Gyula Fehér feher@noserv.obuda.kando.hu

1

INTRODUCTION

What is distributed processing?

Surprisingly, there are many answers to this question

- Client/server systems;
- systems that distribute various processes across a network for execution;
- parallel processing systems which distribute various processes among their own processors;

In April 1992 Andrew Tannenbaum circulated a list of *what he believed were generally accepted truths about distributing computing* to comp.os.research, (a mailing list devoted to topics in operating system research). The result was over *one hundred responses from leading researchers, violently disagreeing with both Tannenbaum and each other!*

Early applications:

- Network file systems,
- Automated teller networks,
- Travel reservation systems,
- Military command and control systems.
-
-
- information management,
- office automation,
- concurrent engineering,
- decision support,
- process control,
- collaborative workgroups.

Motivations

Distributed architectures let users of individual, networked computers

- share programs and data resources (often between geographically remote sites),.
- enhance availability,
- enhance reliability,
- enhance performance.

The goal of a distributed system

is to *hide from applications the fact that resources may be scattered* across a number of hosts.

- Some systems are just a *simple library of routines intended to make communication* between hosts easier, while other systems
- *link the various hosts so tightly together that the applications effectively see one system.*

The architectural extremes are sometimes referred to

- loosly coupled systems
- closly coupled system

Basic components of distributed systems

- ***Programs and data resources*** dispersed across independent computers connected through a
- ***Communication network***
- ***Software*** serves as the glue that holds together distributed systems
    - **The operating software** controls and allocates resources
    - **Application software** does the work.

Communications paradigms

Similarity among various distributed systems
nearly all of them are implemented using one or the other of two communications paradigms:

- ***Remote Procedure Call* (RPC);**
- ***Distributed Shared Memory.***

These classifications are notable because of an important, but often misunderstood, paper by *Lauer and Needham* [1978],

- shared memory and message passing are semantically equivalent
- (on a single system) shared memory and message passing ca have euqally good performance.

The importance of Lauer and Needham's result
- programmers can choose whichever paradigm they prefer (message passing or shared memory),
- confident that neither scheme allows them to solve a problem the other does not.

Communication Model

- ***If the computational pattern of an application*** (modelled by graphs in which processes are represented as nodes, and communication links as edges) ***is isomorphic to the underlying physical communication network topology, everything works fine.***
- However, the physical neetwork is usually in the form of an irregular topology. Hence, we introduce an ***intermediate (virtual) regular topology*** that can be embedded into an irregular-shaped physical network.
- ***It is necessary to hide network complexity from users***. The idea is to introduce a hierarchy of control in the network environment.

Communication system

- The interrelationship between *application needs complex communication services*.
- **Existing** distributed systems are based on packet-or circuit-switched networks with *slow, unreliable, fixed-bandwidth communication channels.*
- **Emerging** local, metropolitan, and wide area networks have transmission speed to over 1 Gbps providing *sufficient bandwidth* for many distributed high-performance applications.
  - These developments, along with *recent standardization* efforts, are stimulating research into a new generation of distributed systems that use advanced computing technologies to provide high-performance computing environments,
  - *Communication channels would no longer be the bottleneck* of a distributed computation, since large amounts of data could be transferred quickly and easily.
  - Fundamental *imitation due to the speed of light* in fibre.
  - *Complex services* (packet-switching, data transfer, on-demand virtual circuit assignment, multiple logical channels, guaranteed qualities of service, and network management *are provided directly by the physical network level.*
  - *The system level handles the management of all application-specific virtual topologies.* (The system level is a logical view of the physical level characterised by a regular topology projected onto the physical network.)

The requirements of new applications

Distributed Multimedia Information Sytems

- **Recent developments in high-speed communication technology** have produced new applications for distributed computer systems that require multiple media types. **Multimedia applications** include office automation, computer-aided design, video teleconferencing, electronic collaboration. These applications require

    - high throughput,
    - guaranteed response times,
    - potentially large work groups,
    - a mix of PCs, workstations, and servers, and
    - interoperability with other networks.

High-performance Computing.

Offers great potential for continued rapid growth in the processing power

- high throughput,
- resonable response times,
- a small number of collaborative users (like workgroups),
- high-end workstations and graphic servers, and
- interoperability with other networks.

Coordination Models

- A coordination model establishes *logical roles* (along with associated behaviours for applications that assume such roles) *for executing distributed interactions*.
- Roles and behaviours *specify who can initiate interactions, who can respond, how to retrieve results (if appropriate), how to handle errors, and so on.*

The client/server coordination model

- Offers *simplicity* in closely matching the flow of data with the control flow,
- Promotes modular, flexible, and *extensible system designs*.
- *Any* data *resource* or computing operation can be organised, integrated, and accessed *as a service*
- Services include *operating system functions* (such as naming and authentication), shared information resources (such as network file systems and printers),
- *applications* (such as data base engines, electronic mail, and dedicated graphic-or numeric-compute engines),
- *Programs can act as eihter clients or servers*, depending on the context.
  For example, a **file server** that invokes an authentication service before releasing requested files (to application clients) represents a resource server acting as a client to obtain access control services from a system server.

A major limitation in the basic client/server model

- **Its focus on clients requesting individual services**. Clients often need to invoke multiple services.

Requesting Services from Multiple Servers

- One *design variable* for applications where clients require multiple services is *the way clients locate and interact with servers.*
- *Small, static client/server applications tend to support a few distinct services*, such as file, print, and database sharing.
- *Tracking available services* and the identities and the locations of the servers that support them *is relatively staightforward.*
- But this *`hardwired' strategy breaks down for larger systems* that evolve incrementally
- *Access problems become particularly acute in heterogeneous systems,* where client interfaces for services may differ radically across different servers and platforms.
- Consequently, developers and their *client programs require system-level support to identify suitable servers* and obtain services from them.

Managing Requests for Multiple Services

Explicit request: the client issues *distinct requests for each desired service* and for any processing required.
Implicit request: the client issues a single request to a server application. This server is complex in that it interacts with other servers to execute the client's request. For example, a heterogeneous data base system may

- map a high-level request for information into subqueries to independent, networked databases.
- apply a relational-join operation to merge the replies; and
- return the resulting compound data set to the client user.

Receiving Responses to Service Requests

The behaviour of a client or complex server in receiving responses to service requests *depends on the underlying communication model* :

Network communication can be

- blocking or
- non-blocking.

For example, *a process that envokes a blocking network input/output operation*, such as reading from or writing to a remote file, *suspends until that operation is complete*.

- Blocking effectively *synchronizes* the source and target processes (inefficient).
- Conversely, non-blocking communication is *asynchronous*; the calling process can handle other tasks, pending completion of requested operation (control complexity).

Critical design issues

- locating programs and data resources distributed across the network,
- establishing and maintaining *interprogram communication on the network*,
- coordinating the execution of distributed applications,
- synchronising replicated programs or data to maintain a consistent state,
- detecting and recovering from failures in an orderly, predictable manner,
- securing resources by limiting remote access to authorised users.

COMMUNICATION SYSTEMS ARCHITECTURES

Topology

Shared-medium topologies (for example, rings and buses)

- Distributed, peer-to-peer environment (classical LANs).
- Attractive because they provided sufficient bandwidth at low cost.
- The combination of peer-to-peer relationship and shared-medium required more complex access protocols (CSMA/CD and token-passing)
- In a shared-medium topology, the bandwidth of a single link is divided among all the stations.

Future applications will require LANs with more bandwidth per station than is now feasible through shared-medium topologies.
Switch-based LANs - such as synchronous-transfer-node (ATM) LANs and switched Ethernet - *are attractive because they provide aggregate bandwidth*, which is a multiple of the bandwidth of a single link.

Transmission Latency

- The intrinsic latency of all LANs is 200,000 km/sec, corresponding to a propagation delay of *5 nanoseconds/metre*, that is, 50 microseconds at 10 kilometres.
- Interconnect *architecture affects latency* because of the access protocols needed to mediate simultaneous transmission requests.(CSMA/CD, token-ring)
- *Software* that implements the applications and higher layers of the communications protocol stack.
- In the switch-based topologies, the latencies in the MAC-layer function will become shorter.

Single-hop Versus Multi-hop Service

- *Intermediate nodes* separate the destination from the source
- Usually necessary to *choose among several potential transmission routes.*
- *Congestion control, is independent from flow control*; the latter relates only to the point-to-point traffic between source and destination.

Connection-oriented Versus Connectionless Service

From an end-to-end viewpoint

- ***In connection-oriented service***
  - Data cannot be transmitted before **establishing a connection** between the source and destination.
  - Connections deliver **data in sequence**;
  - **Error control** and **flow control** are provided as part of the service.
- ***In connectionless service***
  - **No connection** is established between the source and destination.
  - Each unit of **data contains the destination address** and is transmitted independently.
  - The interconnect makes a `best effort' attempt to deliver the data.
  - Connectionless service **does not guarantee sequential delivery of data.**
  - Nor does it provide error and flow control..

Real-time Constraints of Services

- **Traditional LANs had no real-time constraints**; they emphasized average response time and data throughput.
- **Next generation LANs** (for example, FDDI-II) **provide isochronous service** to meet the *transmission requirements of voice and video applications,* among others.

Fair Access of Services

- The nodes have **equal communications need**.
- **In traditional LANs** the access to the medium is not centrally controlled, **fairness explicitly guaranteed by the architecture through distributed control.**
- As a result, traditional **LAN access protocols**, such as CSMA/CD and token-passing, **are complicated ones**.

Priorities

It may be necessary to *delay some applications* to service time-critical applications.

- The addition of **voice and video communication** to LANs has required the addition of priority mechanisms.
- Since voice and video communication are more sensitive than data and **require isochronous transfer**, the LAN must give voice and video higher priority efficiently.
- **Priorities cannot be hard-wired** on LANs due to the distributed and peer-to-peer nature of the environment.
- **Priority mechanisms must be explicitly included in the LAN protocol.** Examples include token ring and FDDI.

Multiplexing

To exploit increased data transfer rates at longer distances, LANs **need efficient multiplexing** so that the higher bandwidth can be used *to improve interconnect throughput.*

Ethernet and the initial version of the token ring allowed only a single message in the interconnect at any time, **both the token ring and FDDI** now **allow multiple messages**.

Bandwidth Management

- If the interconnect can multiplex data, then **managing the multiple data streams so that each stream gets different shares of available bandwidth** becomes an issue.
- Providing each stream with just the bandwidth it needs **minimizes bandwidth waste** and therefore **improves the cost-to-performance ratio**.
- Bandwidth management is **especially important when the interconnect supports time-critical applications.**

- **LANs did not deal with bandwidth management** before now because applications did not depend on strict time relationships among related messages.
- **It waas sufficient to provide fair access** to all stations.
- The FDDI priority scheme, as an example, allows a given station to rank its traffic in granular fashion, but its architecture still does not allow explicit bandwidth management.

Multimedia applications require bandwidth management to ensure that

- Each information stream gets the bandwidth *necessary to maitain isochronous service* and the
- required information transfer rate.

Next generation LANs, such as FDDI-II
- are incorporating isochronous service and
- bandwidth management (in their architectures).

At present, when such applications are used it is necessary
- to use circuit-switched service and
- to provide end-point bandwidth management withinh each circuit connection.

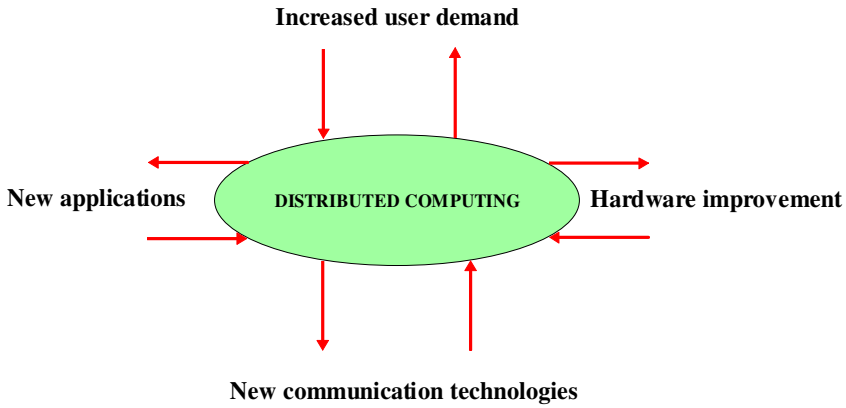
ADVANTAGES OF DISTRIBUTED SYSTEMS OVER CENTRALIZED SYSTEMS

Speed:	***A distributed system may have more total computing than a mainframe***
Economics:	***Microprocessors offer a better price/performance than a mainframe***
Inherent distribution:	***Some applications involve spatially separated machines***
Reliability:	***If one machine crashes, the system as a whole can still survive***
Incremental growth:	***Computing power can be added in small increments***

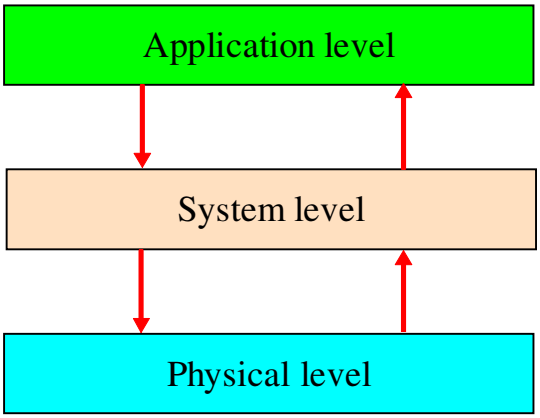
Gyula Fehér feher@noserv.obuda.kando.hu

ADVANTAGES OF DISTRIBUTED SYSTEMS OVER ISOLATED (PERSONAL) COMPUTERS

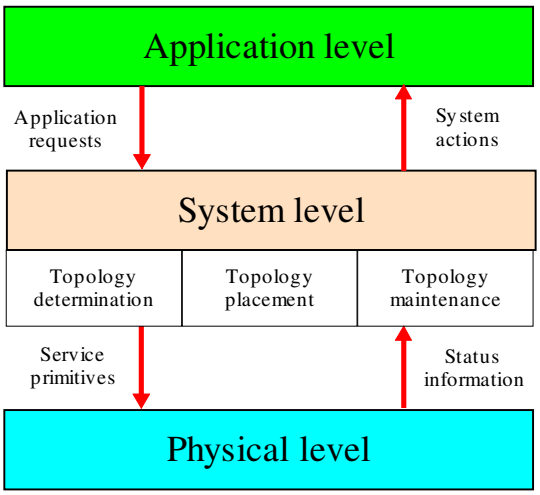
- **Data sharing**: *Allow many users access to a common data base*
- **Device sharing**: *Allow many users to share expensive peripherals*
- **Communication**: *Make human-to-human communication easier, for example, by E-mail*
- **Flexibility**: *Spread the workload over the available machines in the most cost effective way*



TYPES OF CHANGE THAT STIMULATE THE EVOLUTION OF DISTRIBUTED COMPUTING TECHNOLOGIES

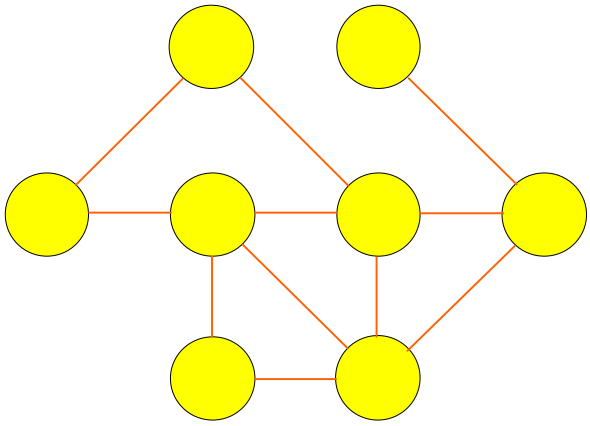


The hierachy of layers

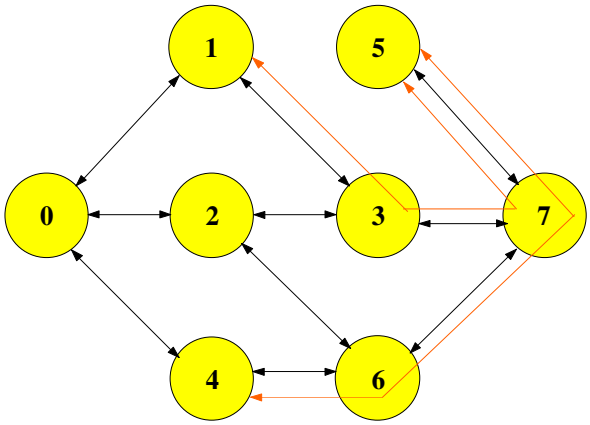


The operation

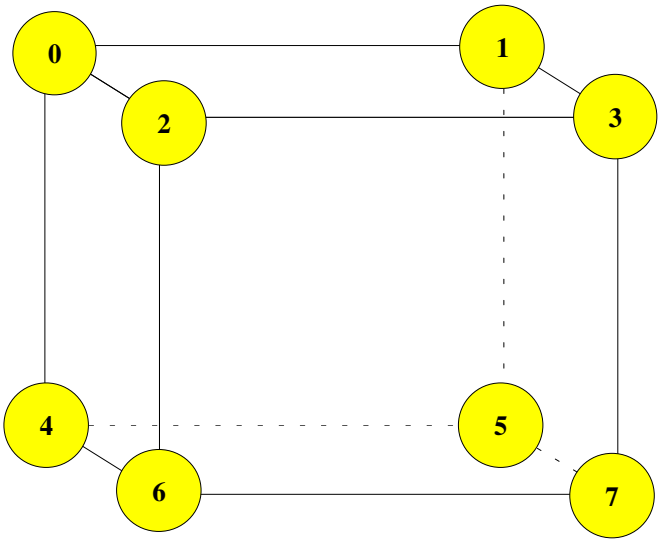
MULTILEVEL COMPUTATIONAL MODEL



Graph with physical graph connections

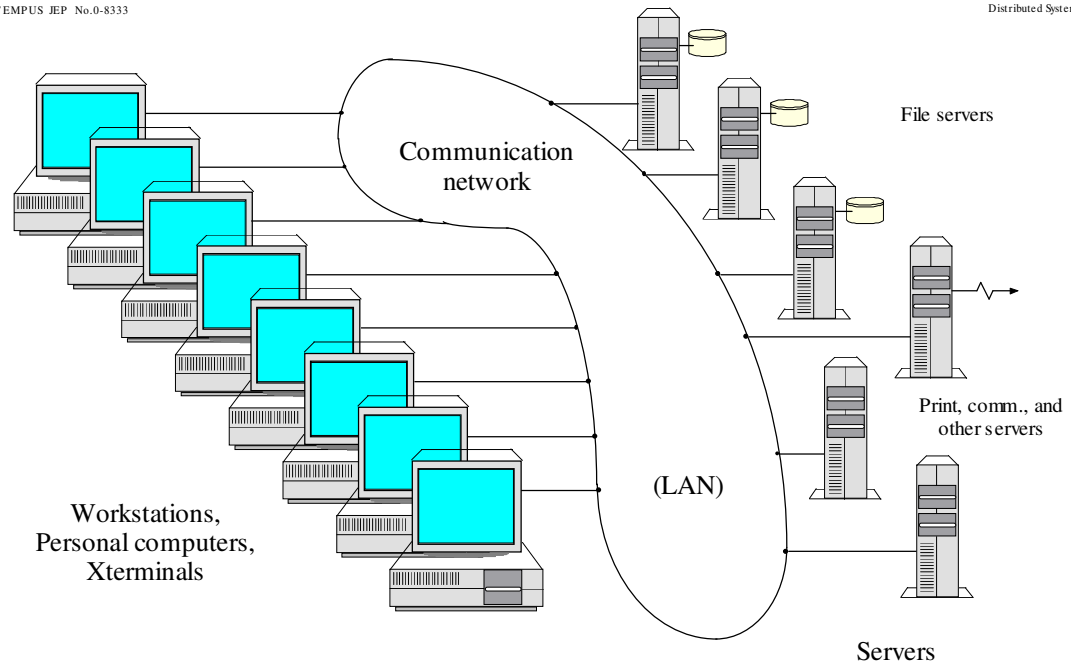


Projected hypercube



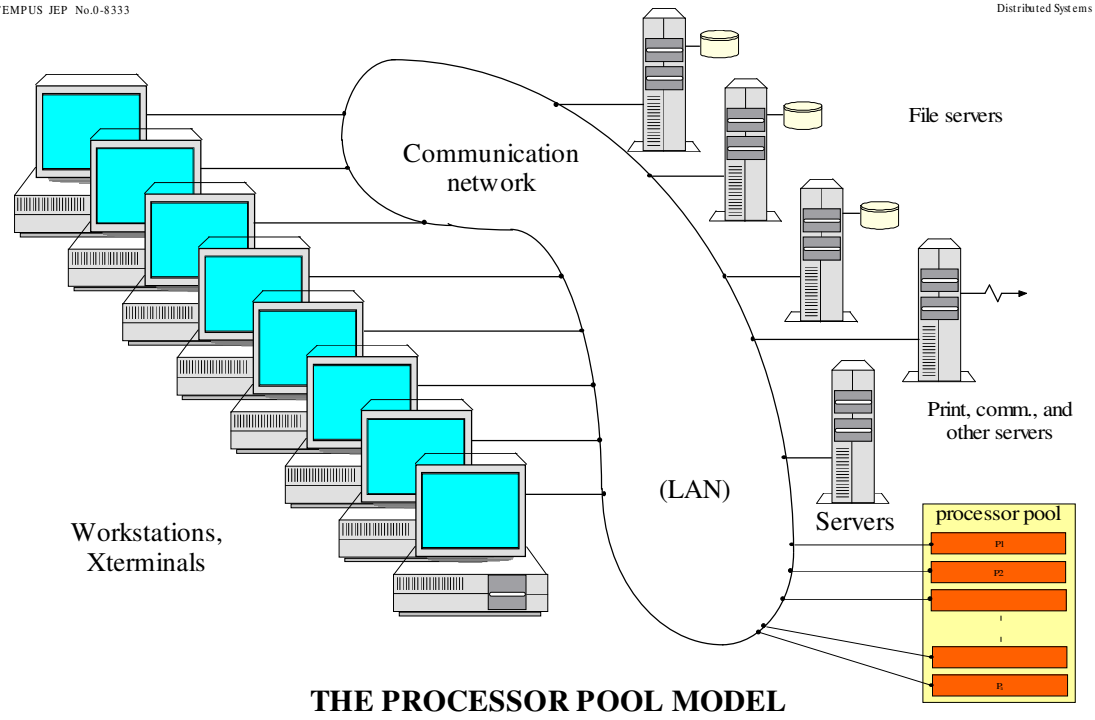
Irregular graph embedded in a hypercube

Gyula Fehér feher@novserv.obuda.kando.hu



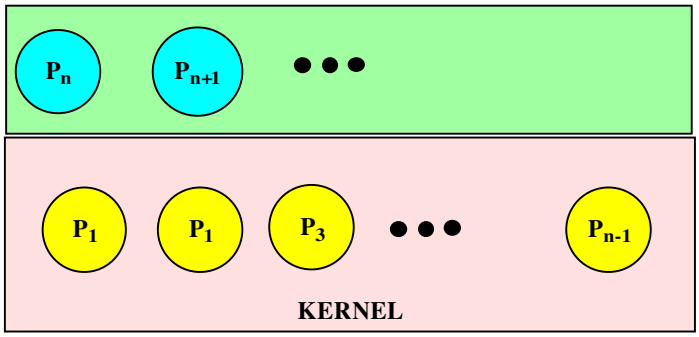
A SIMPLE DISTRIBUTED SYSTEM

Gyula Fehér feher@novserv.obuda.kando.hu

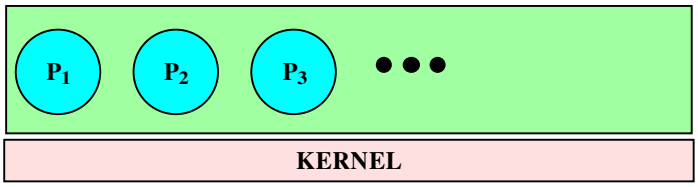


THE PROCESSOR POOL MODEL

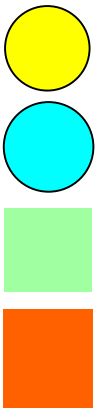
Gyula Fehér feher@novserv.obuda.kando.hu



Monolithic Kernel



Microkernel

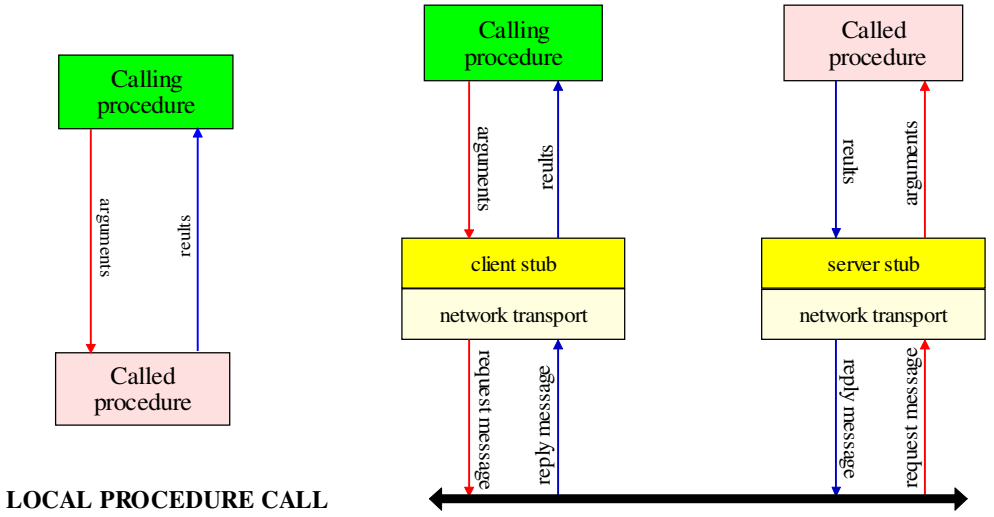


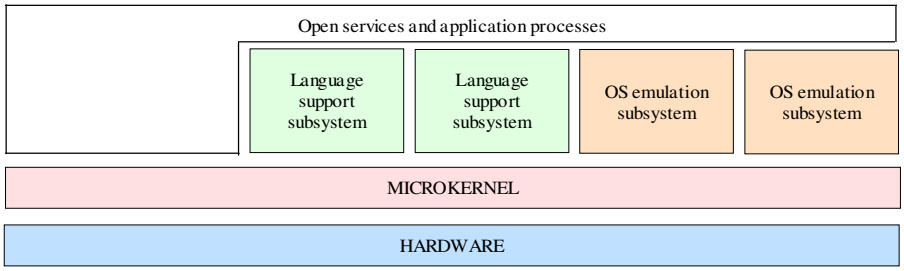
Server process

Dynamically loaded server process

Application code and application data

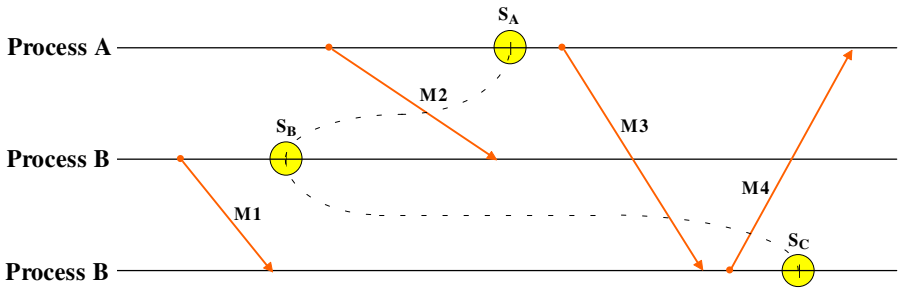
Kernel code and kernel data


Calling procedure
arguments
reults
Called procedure
LOCAL PROCEDURE CALL
Calling procedure
arguments
reults
client stub
network transport
request message
reply message
Called procedure
reults
arguments
server stub
network transport
reply message
request message
REMOTE PROCEDURE CALL

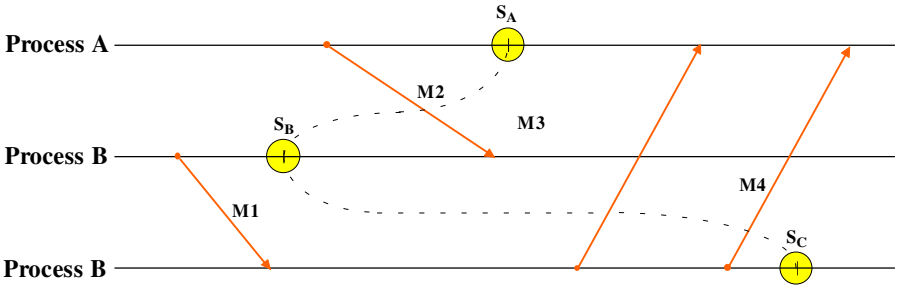


The microkernel layer supports open services and applications

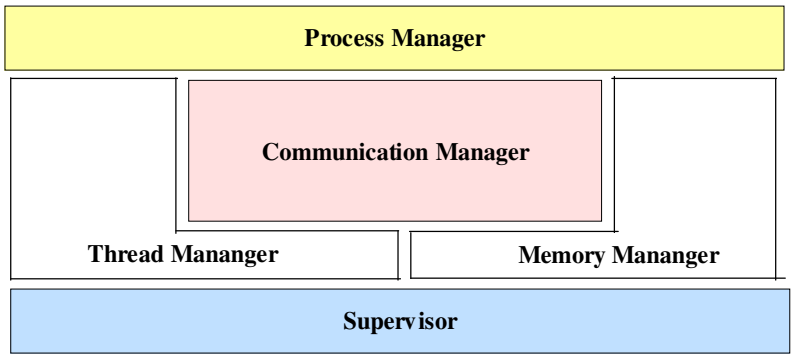
Gyula Fehér feher@noserv.obuda.kando.hu



(a) Inconsistent global state

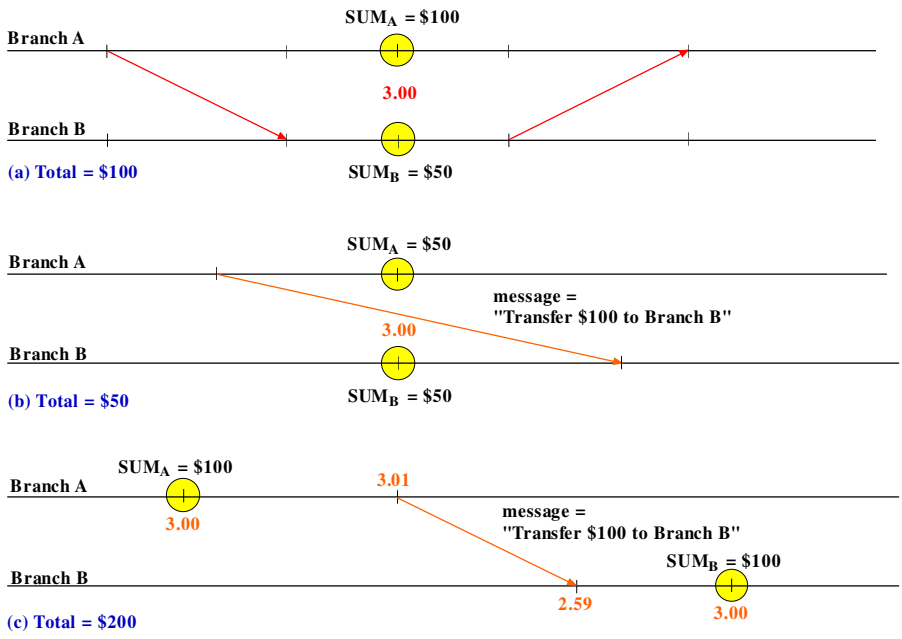


(b) Consistent global state

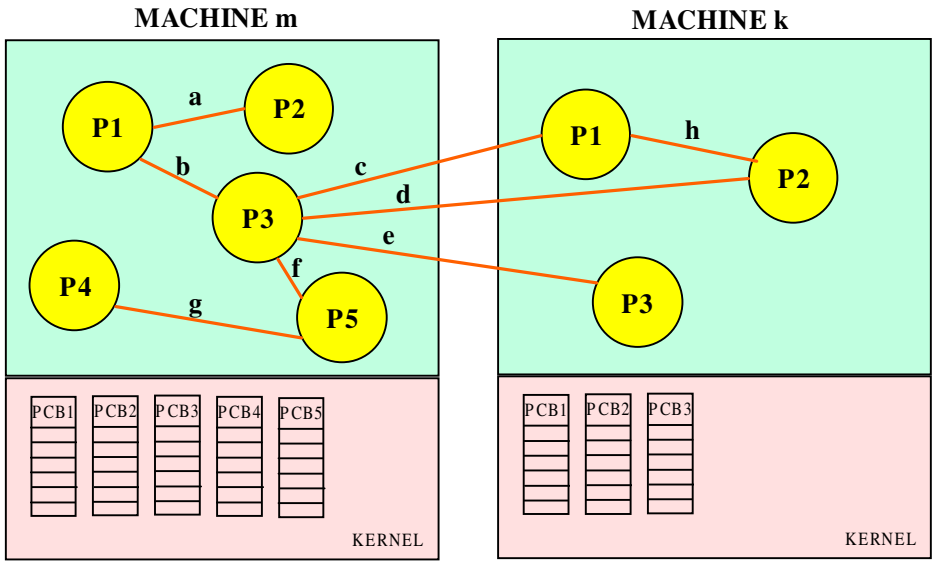


MICROKERNEL ARCHITECTURE

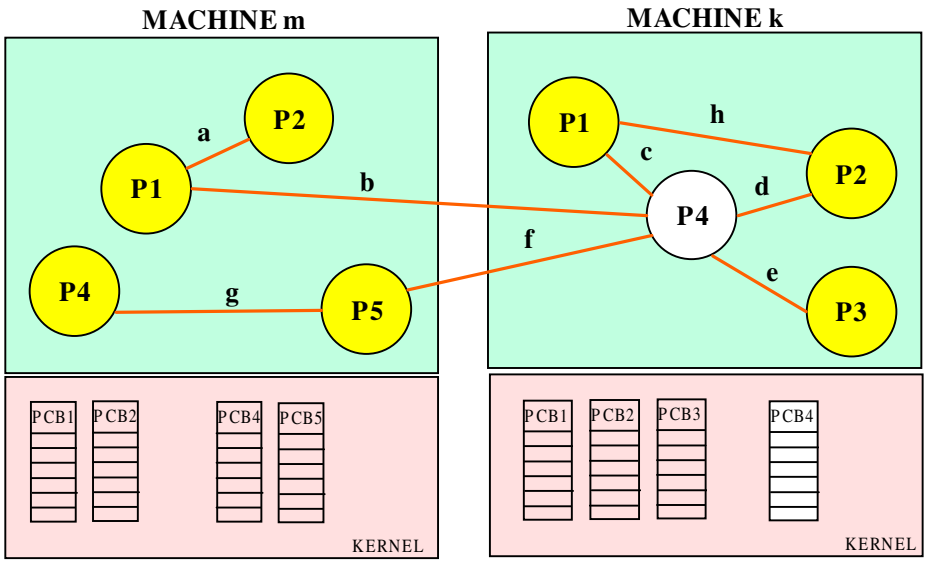
Gyula Fehér feher@noserv.obuda.kando.hu



DETERMINING GLOBAL STATE

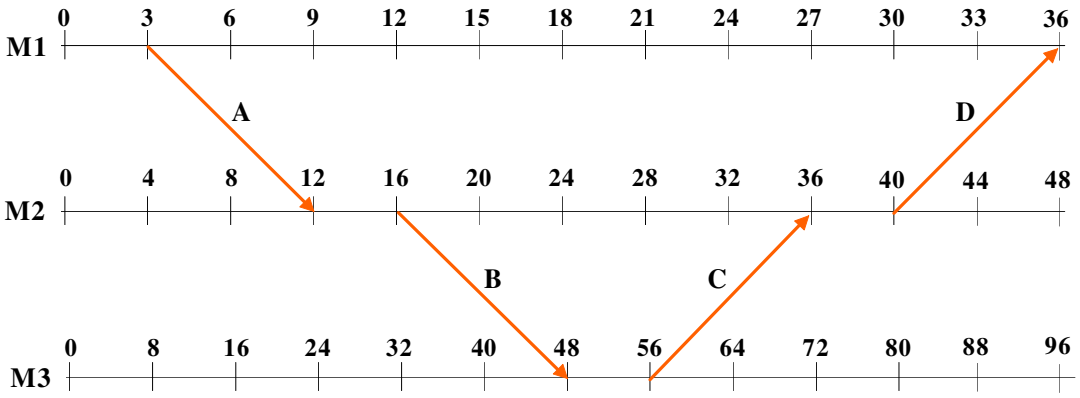


(a) Before migration

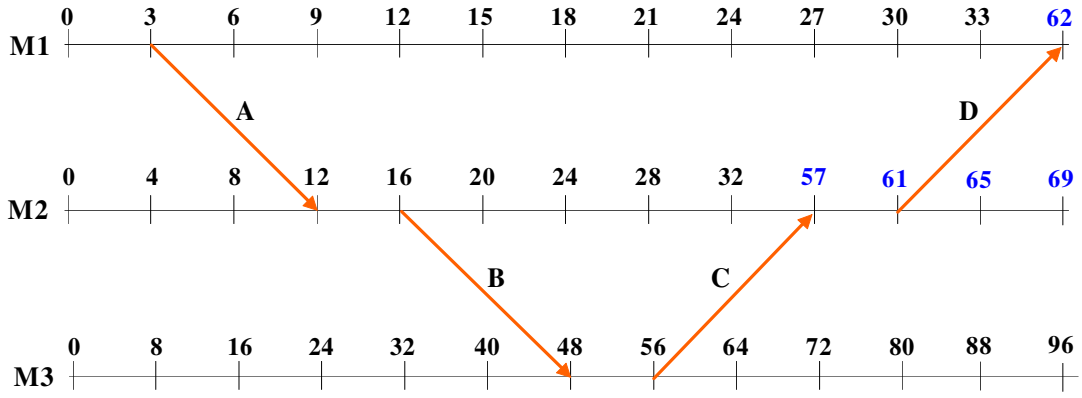


(a) After migration

PROCESS MIGRATION



(a) Three process, each with its clock



(a) Lamport's algorithm corrects the clocks.

CLOCK SYNCHRONIZATION