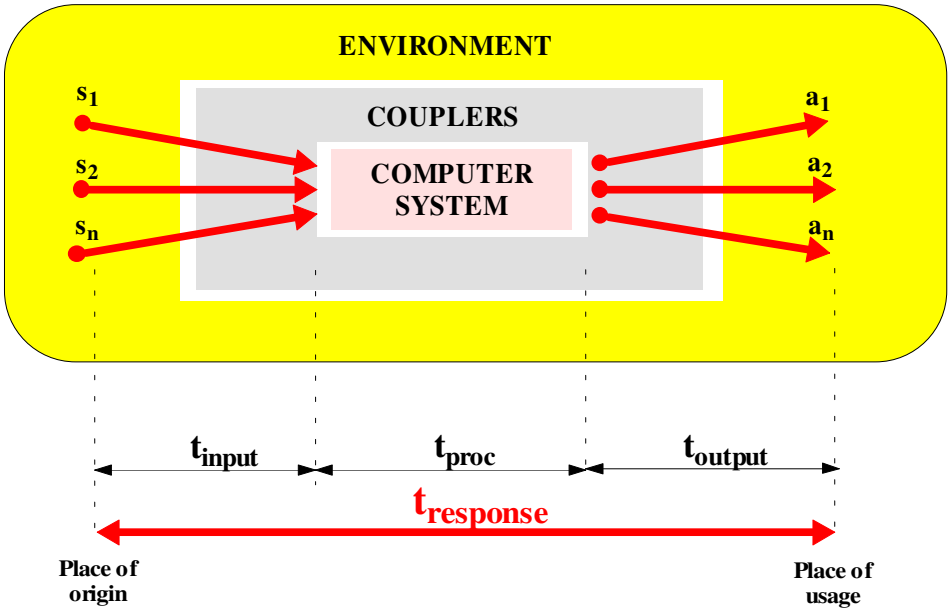
REAL-TIME OPERATING SYSTEMS

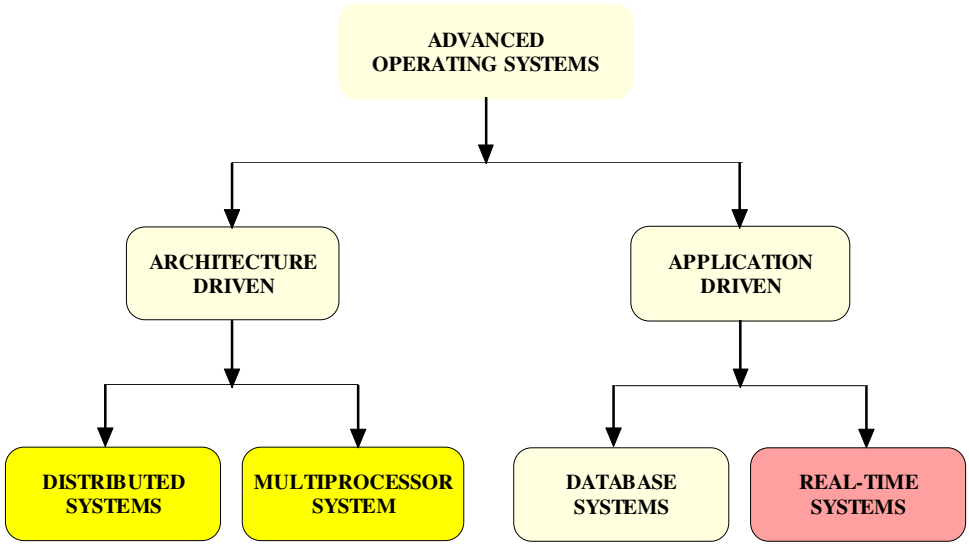
1.

Gyula Fehér feher@novserv.obuda.kando.hu



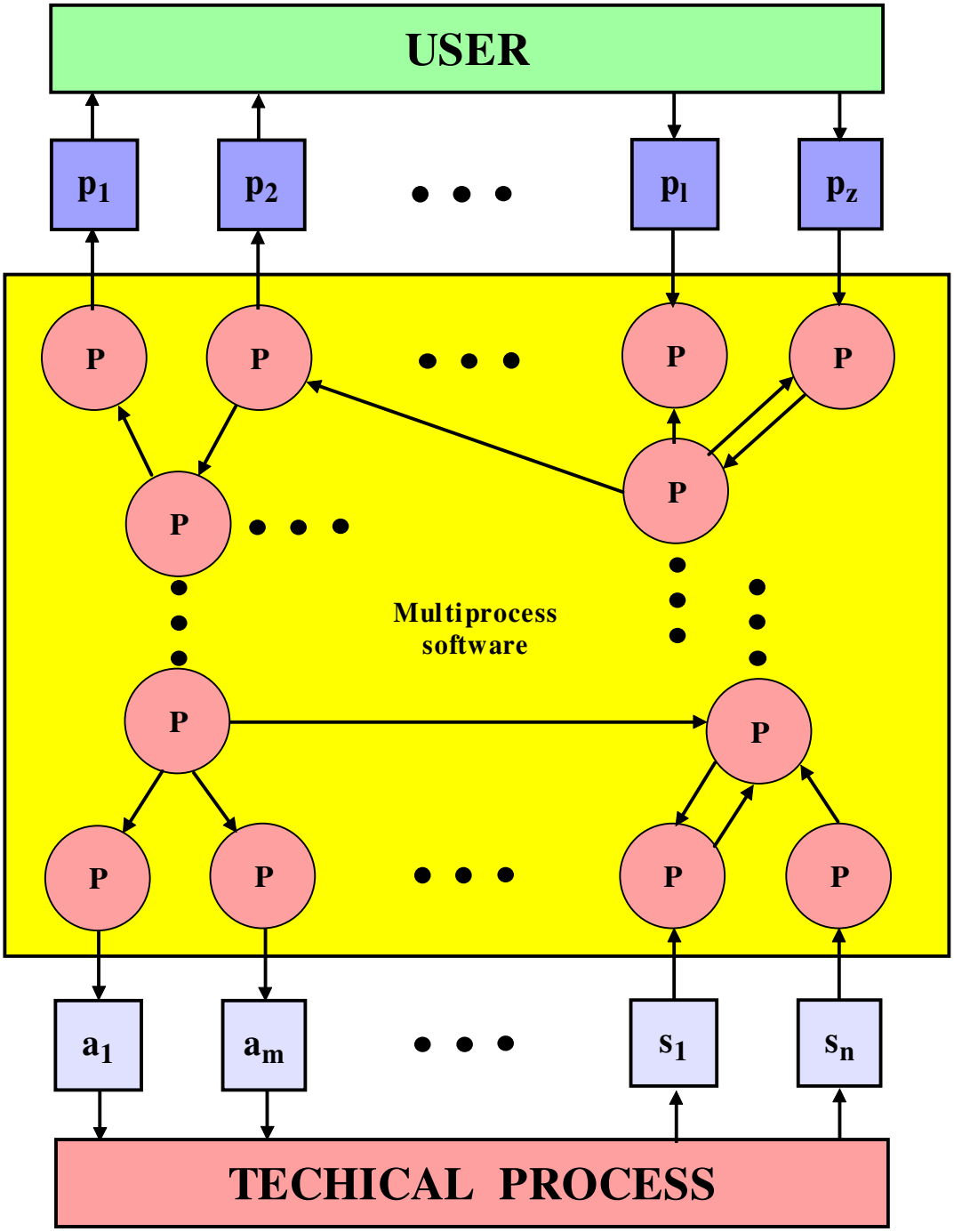
REAL-TIME SYSTEMS
ARE A WELL DEFINED CLASS OF SYSTEMS

Gyula Fehér feher@novserv.obuda.kando.hu



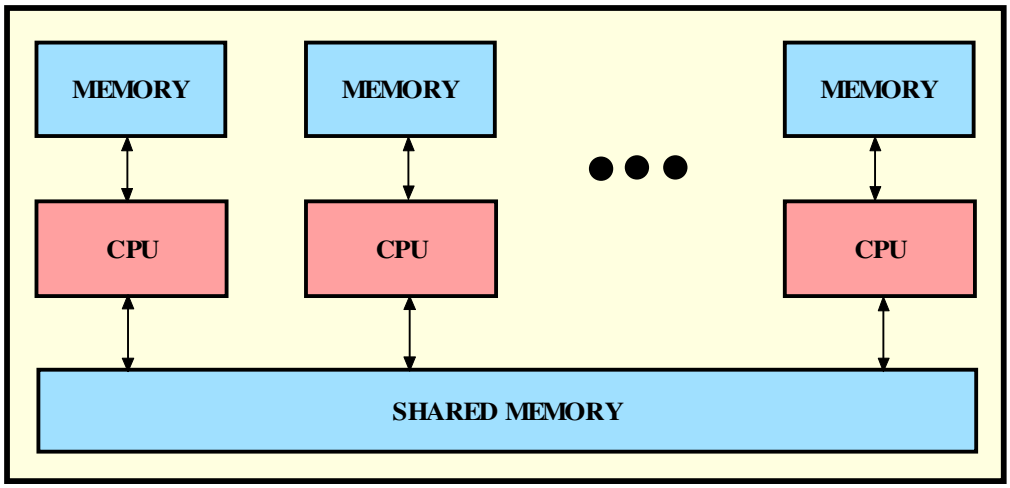
Classification of advanced operating systems

Gyula Fehér feher@novserv.obuda.kando.hu

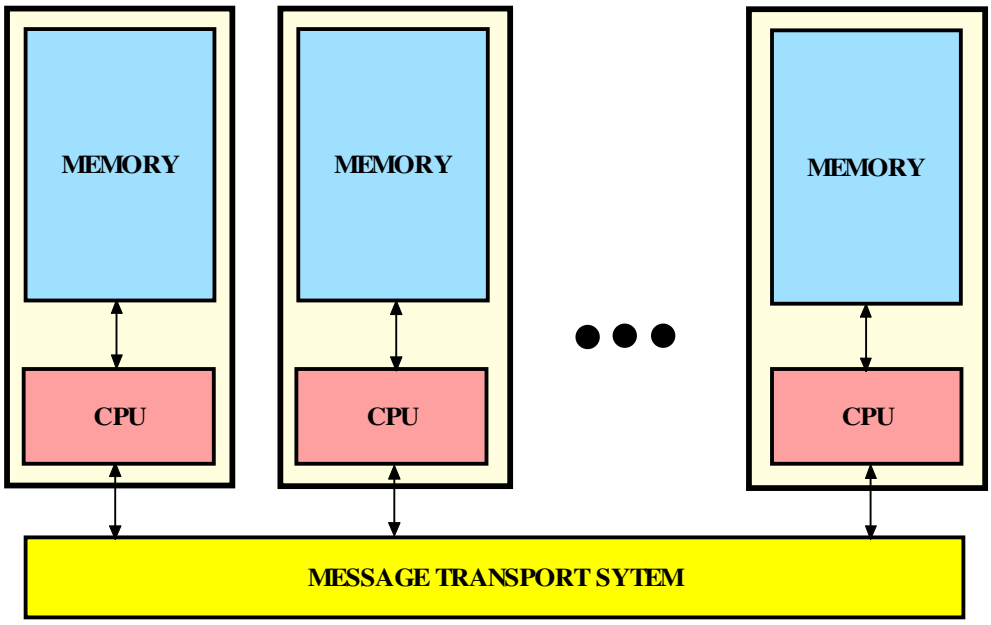


STRUCTURE OF PROCESSCONTROL SOFTWARE

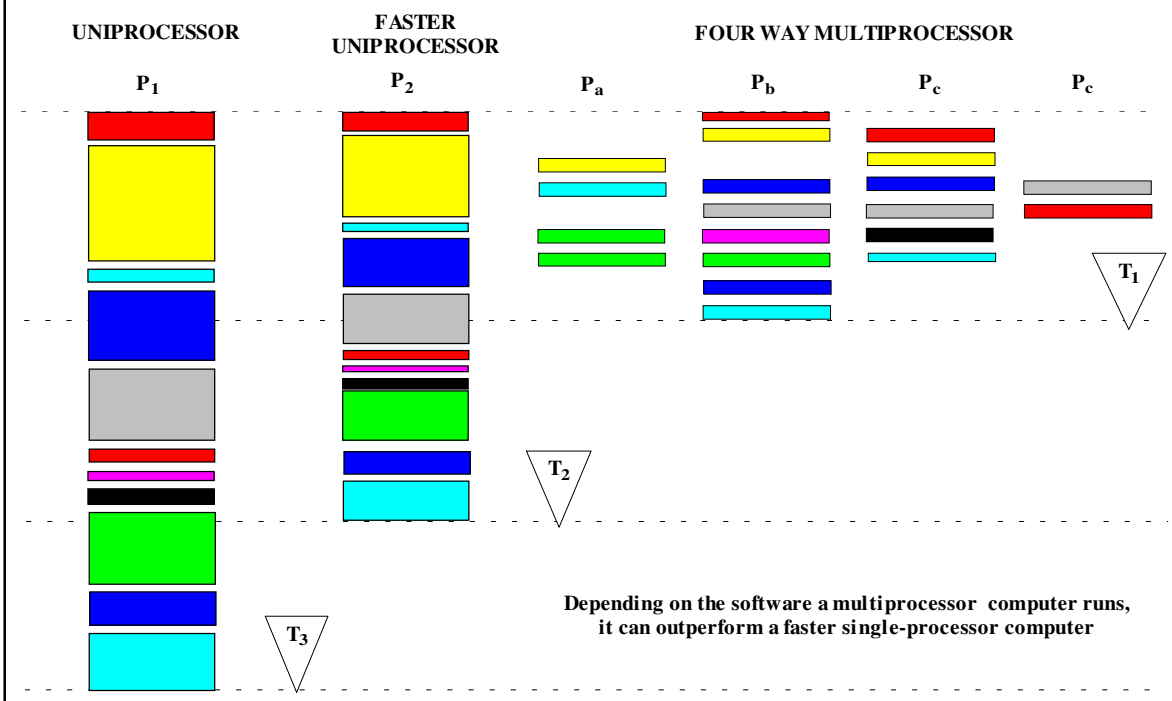
Gyula Fehér feher@novserv.obuda.kando.hu



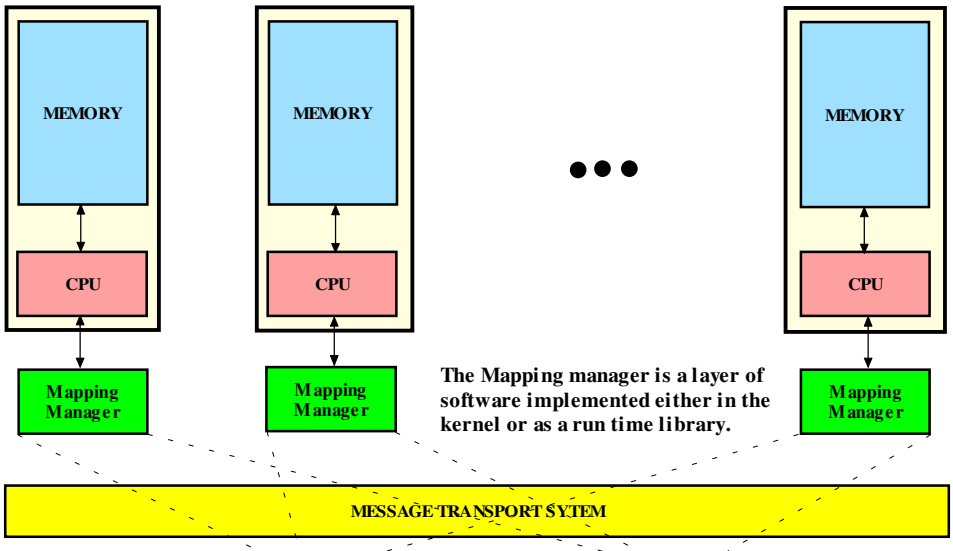
Structure of a multiprocessor system



They have based on the message passing model in which processes interact and share data with each other by exchanging data in the form of messages: Hoar's CSP, the client-server model, RPC.



Gyula Fehér feher@novserv.obuda.kando.hu



The Mapping manager is a layer of software implemented either in the kernel or as a run time library.

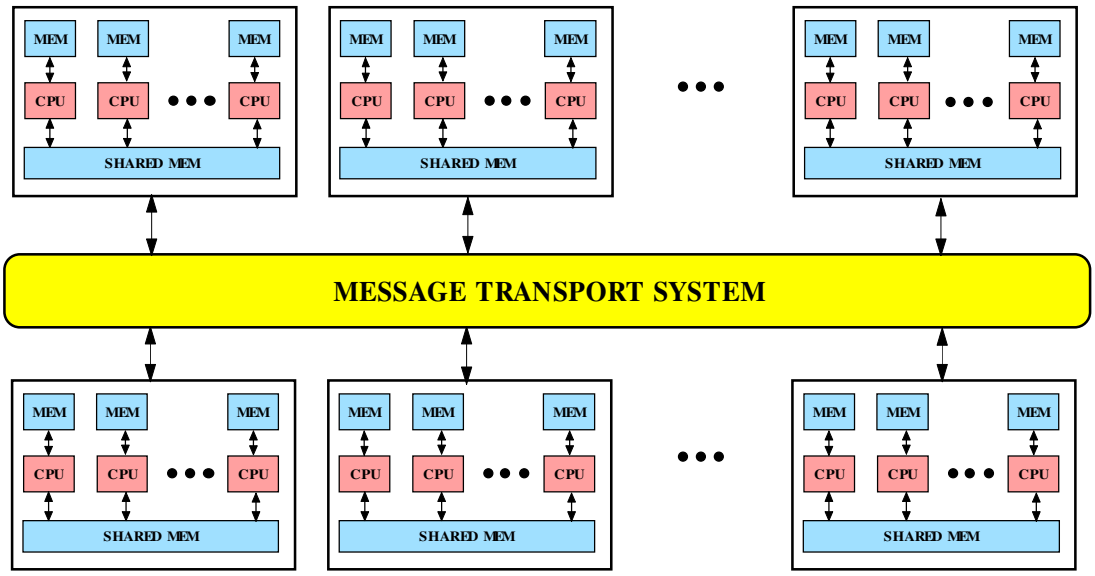
- No explicit Message Massing
- Simpler abstractions for sharing data.
- Cheaper to build then mp. system
- Enormous memory space can be used
- No memory access bottlenack

SHARED MEMORY

Programs access data in the shared address space just as they access in a traditional virtual memory.

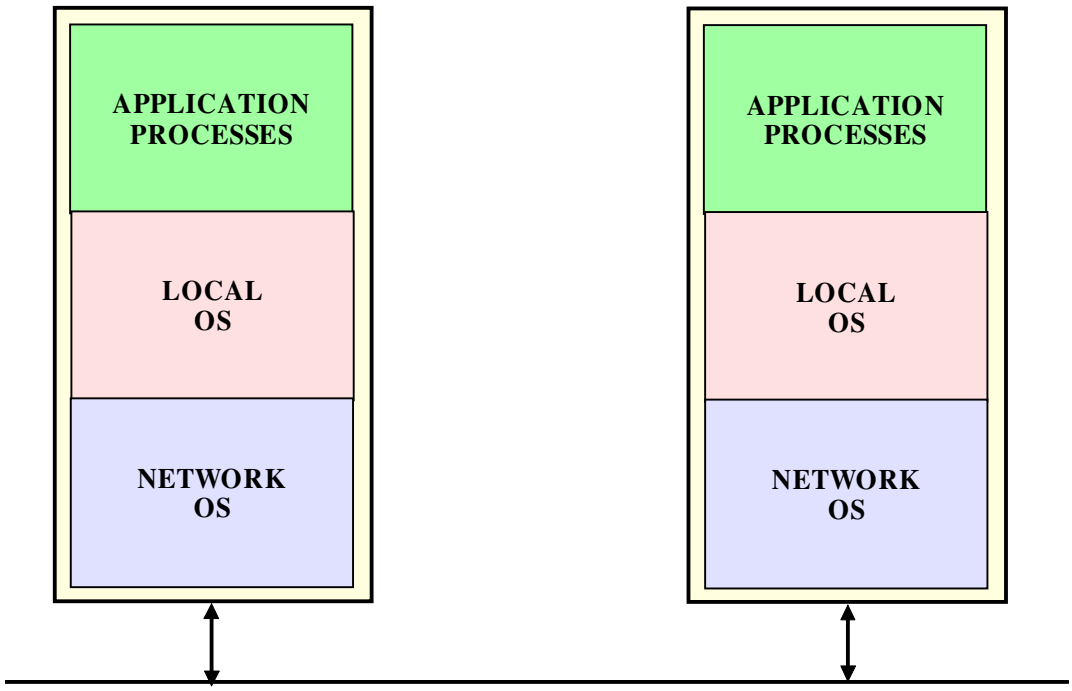
Structure of Distributed Shared Memory System

Gyula Fehér feher@novserv.obuda.kando.hu

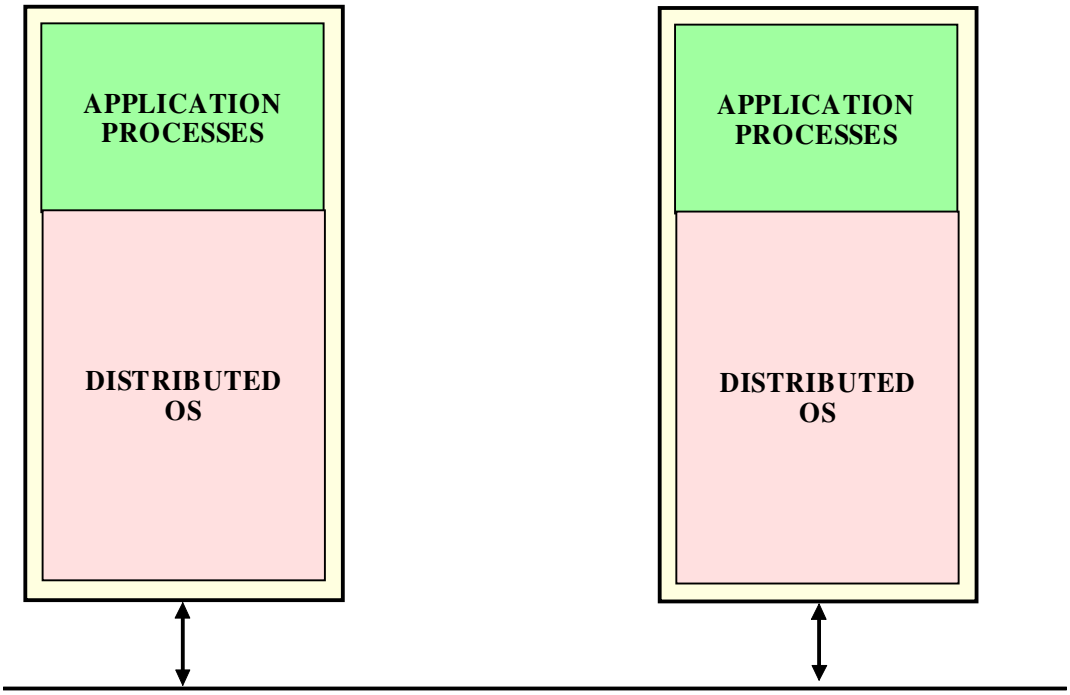


Strusture od distributed multiprocessor system

Gyula Fehér feher@novserv.obuda.kando.hu

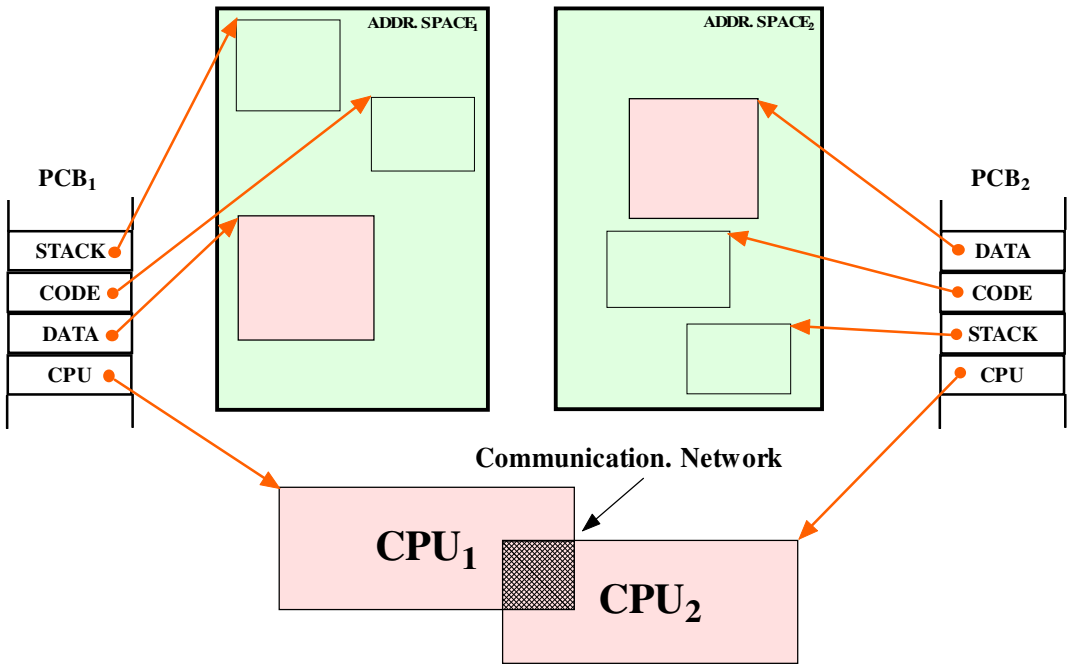


A networked system provides each node with its own local OS and passes node communications through a separate network OS

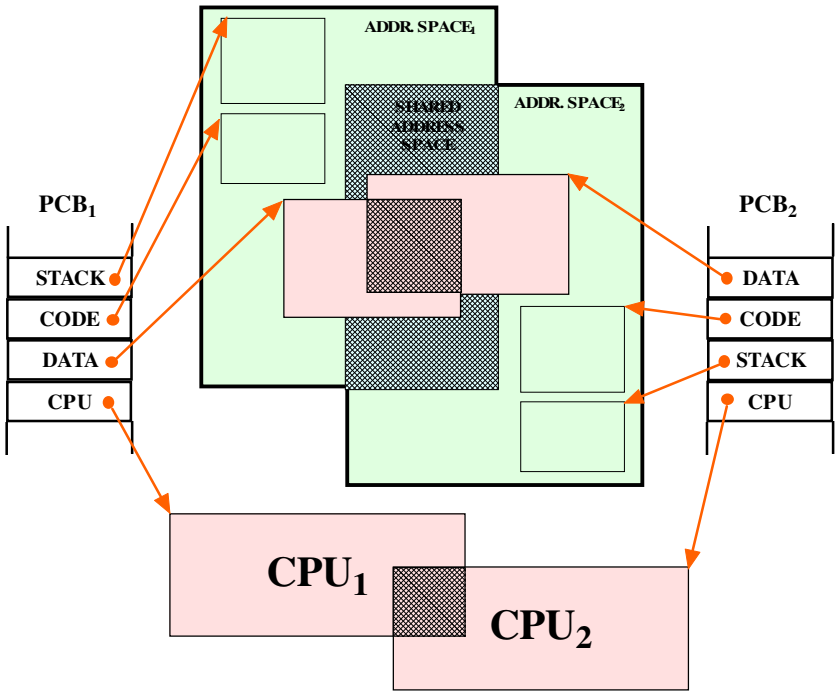


A distributed system provides each node with the OS's microkernel to handle process communication and process management. The OS and the application processes lie scattered throughout the system.

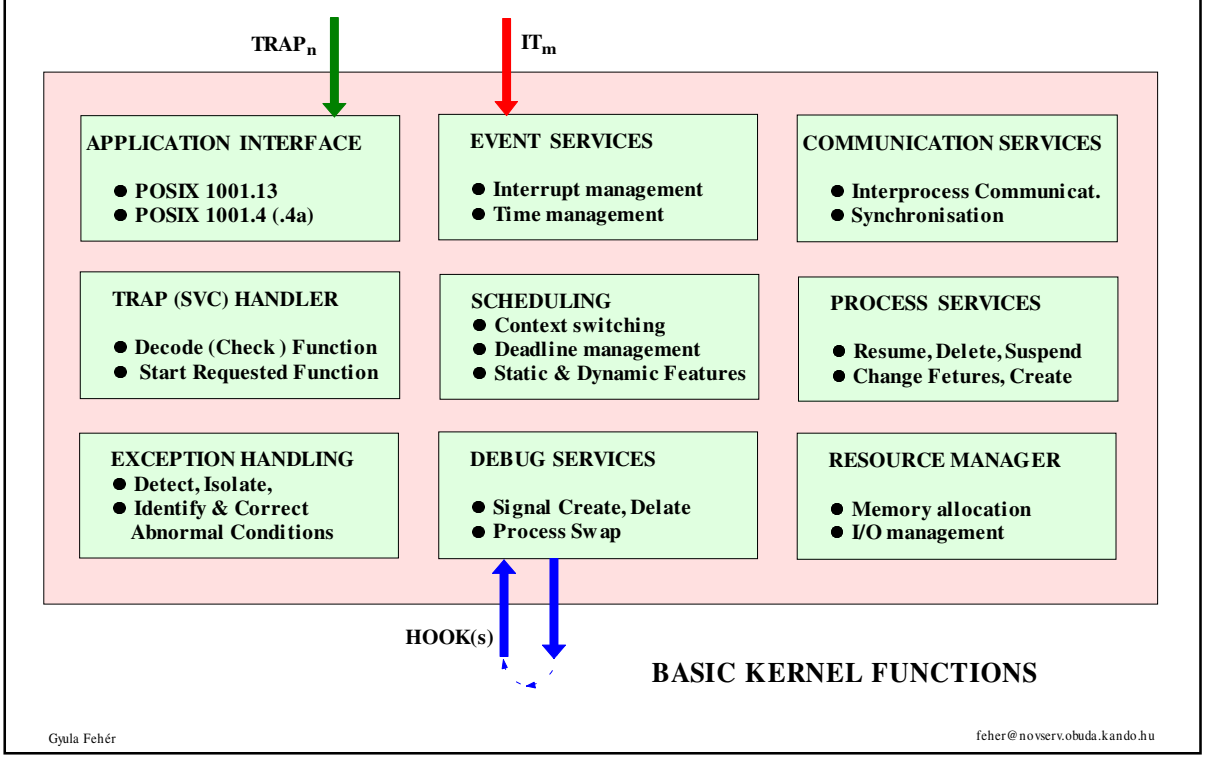
Gyula Fehér feher@novserv.obuda.kando.hu



Gyula Fehér feher@novserv.obuda.kando.hu

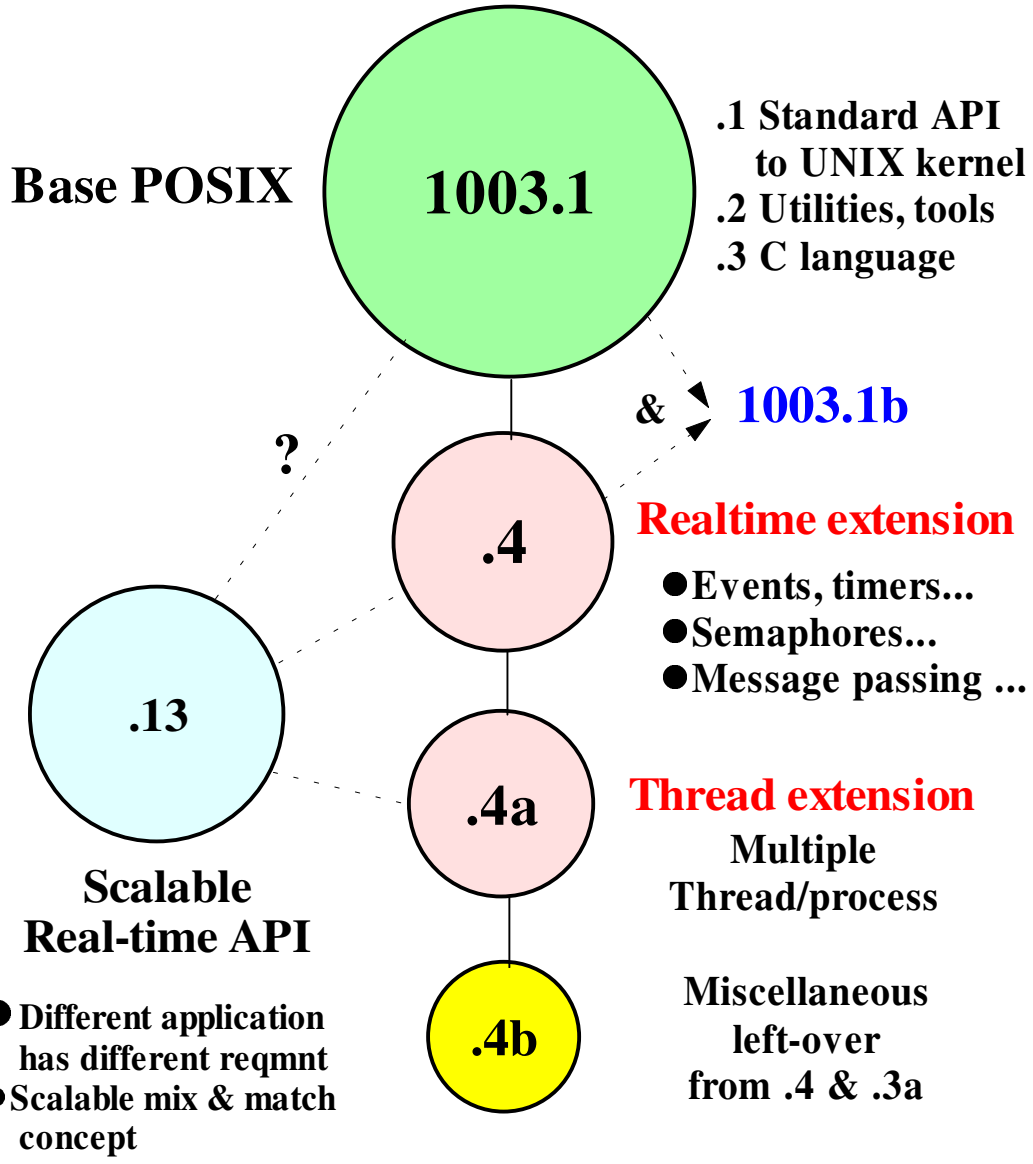


Gyula Fehér feher@novserv.obuda.kando.hu



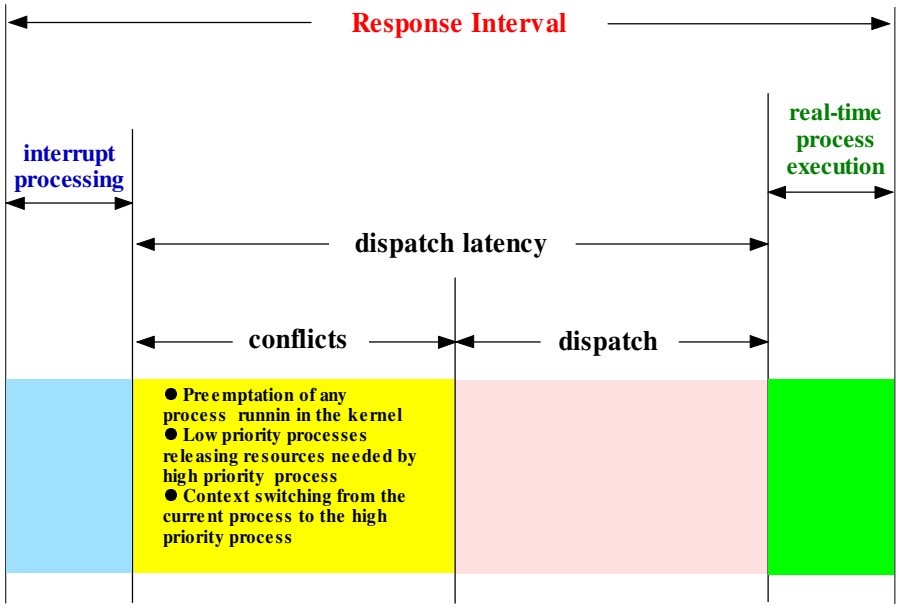
BASIC KERNEL FUNCTIONS

Gyula Fehér feher@novserv.obuda.kando.hu



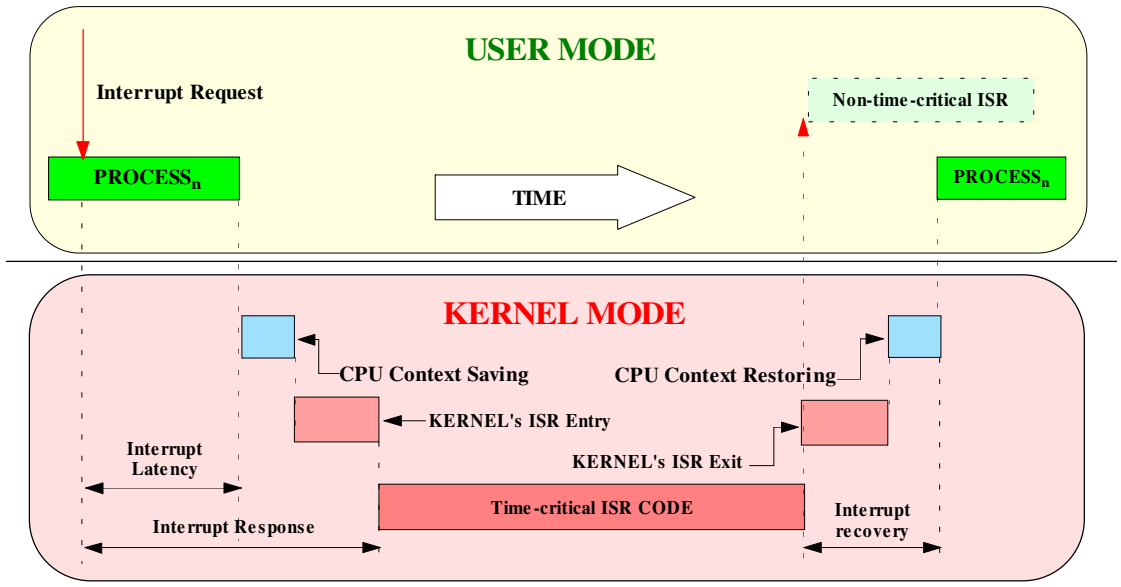
Standardisation of Real-Time API

Gyula Fehér feher@novserv.obuda.kando.hu



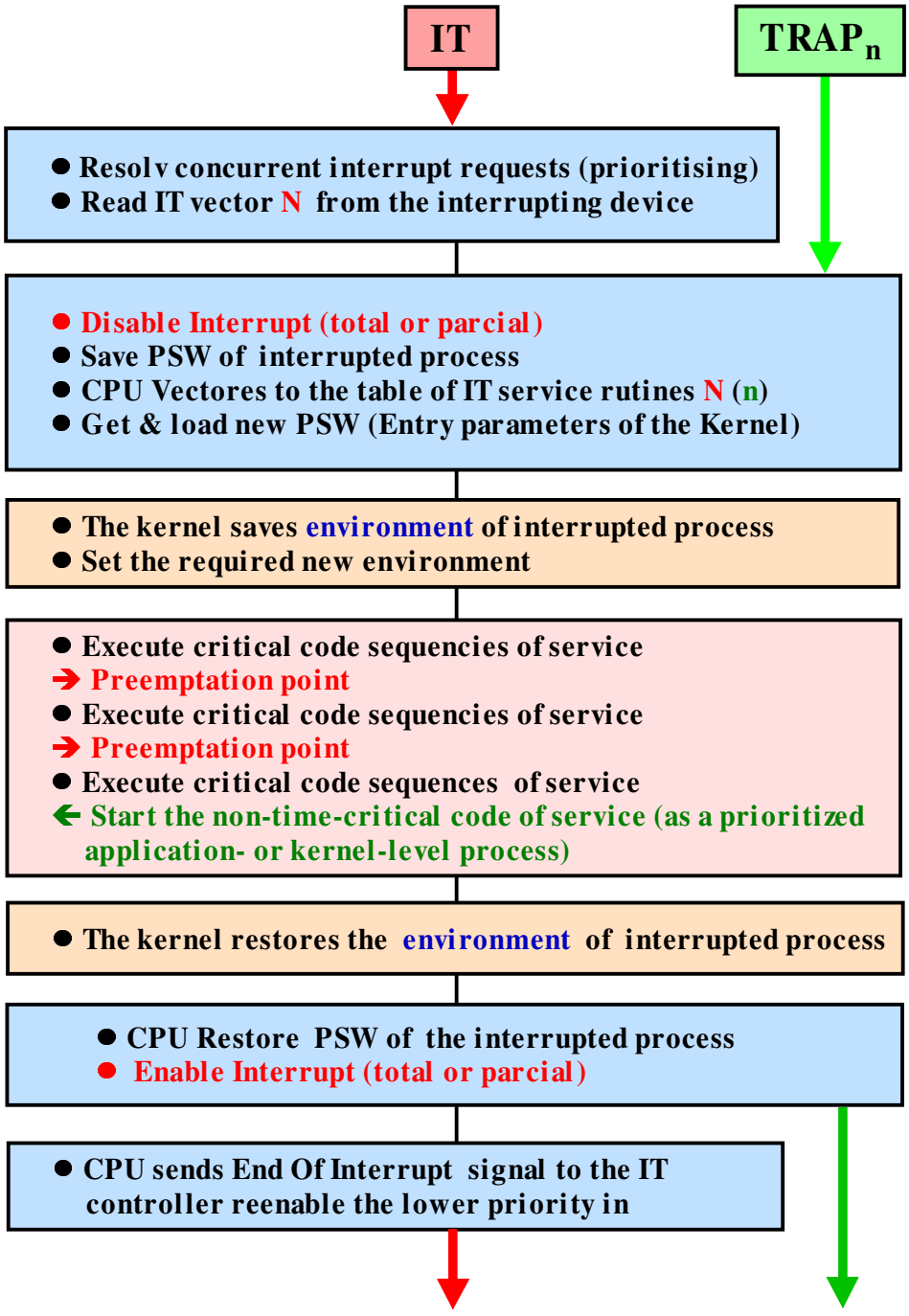
The Components of Response Interval

Gyula Fehér feher@novserv.obuda.kando.hu



Process of an Interrupt

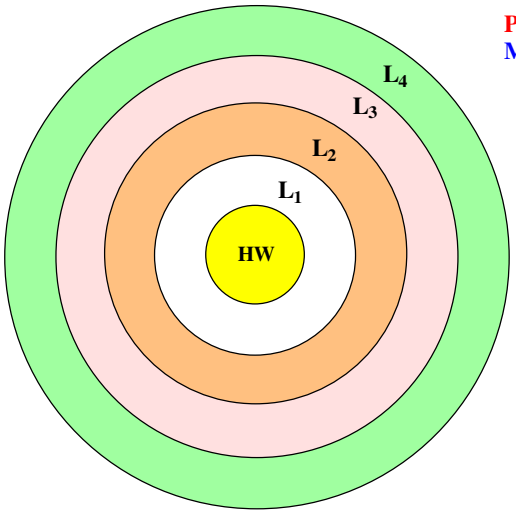
Gyula Fehér feher@novserv.obuda.kando.hu



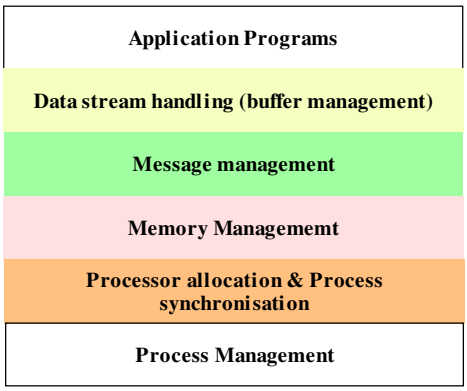
Processing an Interrupt or a Trap

Gyula Fehér feher@novserv.obuda.kando.hu

- Policies are likely to change with time, application „..➔ good OS design must separate policies from machanisms.
- Since policies make use underlaying mechanisms, this separation greatly contributes to flexibility, as policy decisions can be made at a higher level.



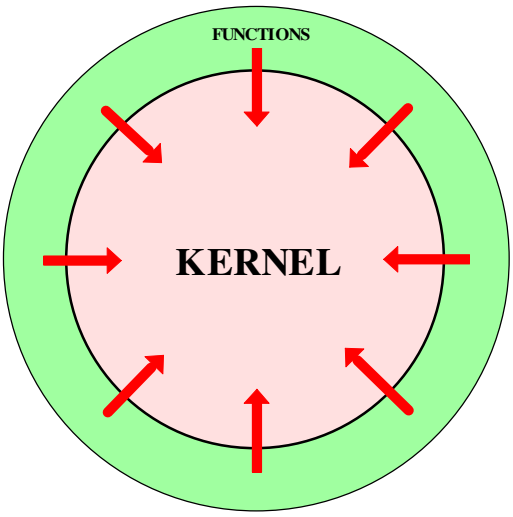
Policies: what should be done
Mechanisms: how it should be done

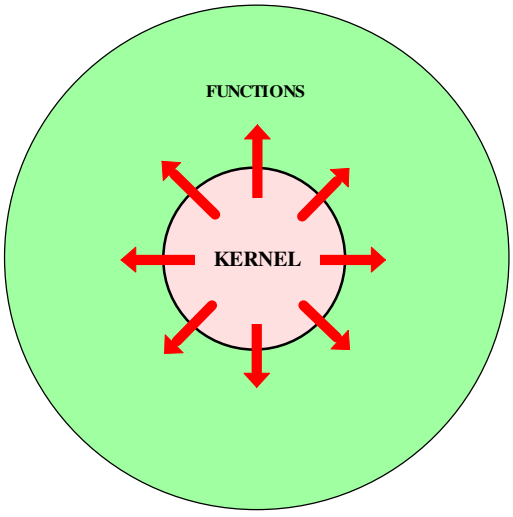


Operating structuring principles: Layered Approach

Gyula Fehér feher@novserv.obuda.kando.hu

Kernel (nucleus): a collection of primitive facilities over which the rest of OS is built. It support only mevhanisms and that all policy decisions should be left to the outer layer. Including too much functionality in it results in low flexibility at a higher level, whereas including too little functionality in it results low support at a higher level.





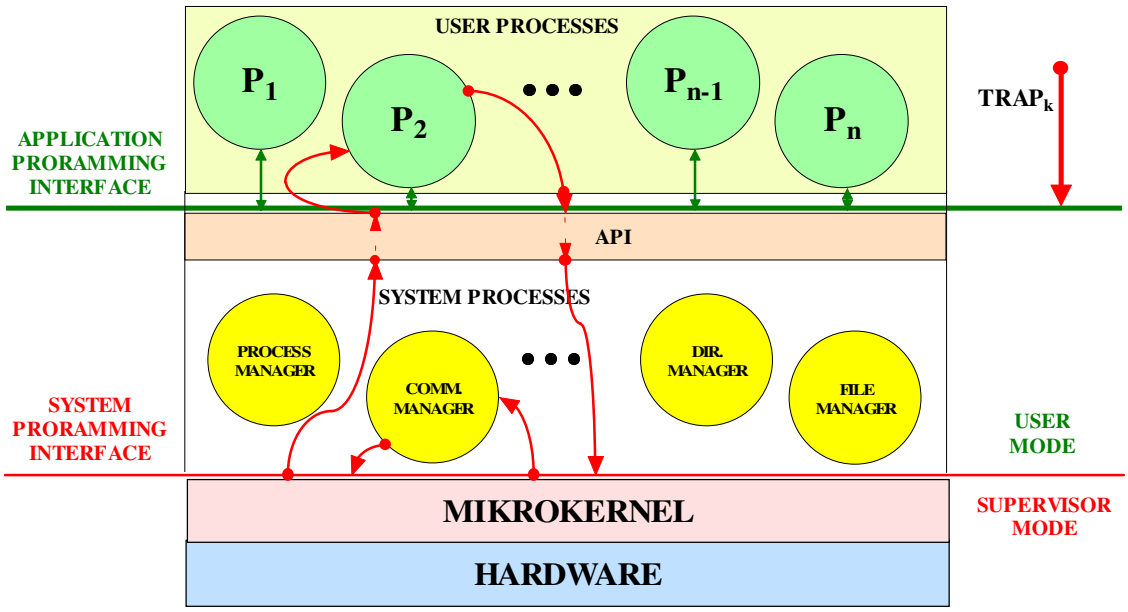
The Kernel performs all basic OS function (efficient)
- It is coded in a non-modular way (closed)
- Because the lack of structure, altering any function is difficult & risky

The Kernel provides only the most essential functions:
- Interrupt Handling
- Low-level I/O & network services
- Interprocessor Communication
- Limited Scheduling & Process management

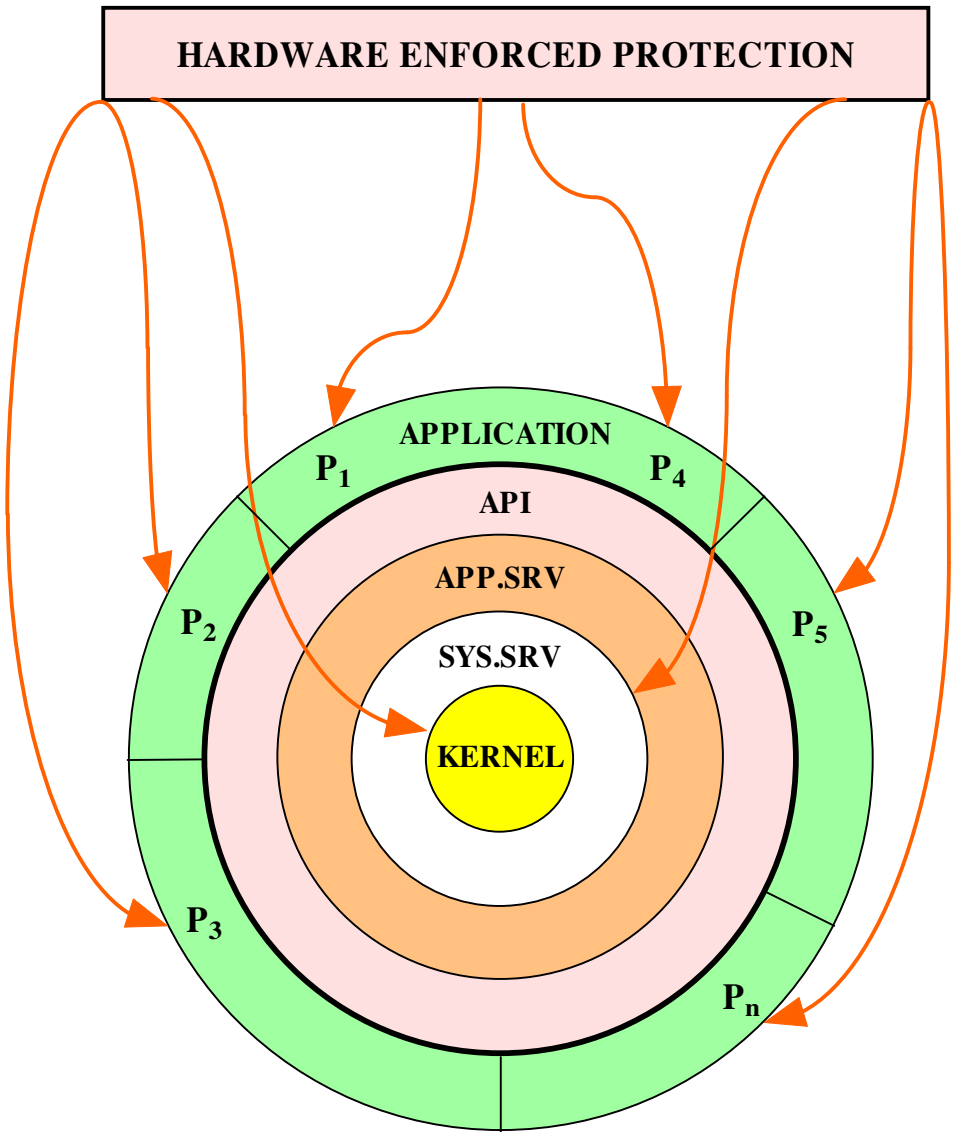
All the other services realised as user-level servers

Operating System structuring principles: Kernel Based Approach

Gyula Fehér feher@novserv.obuda.kando.hu



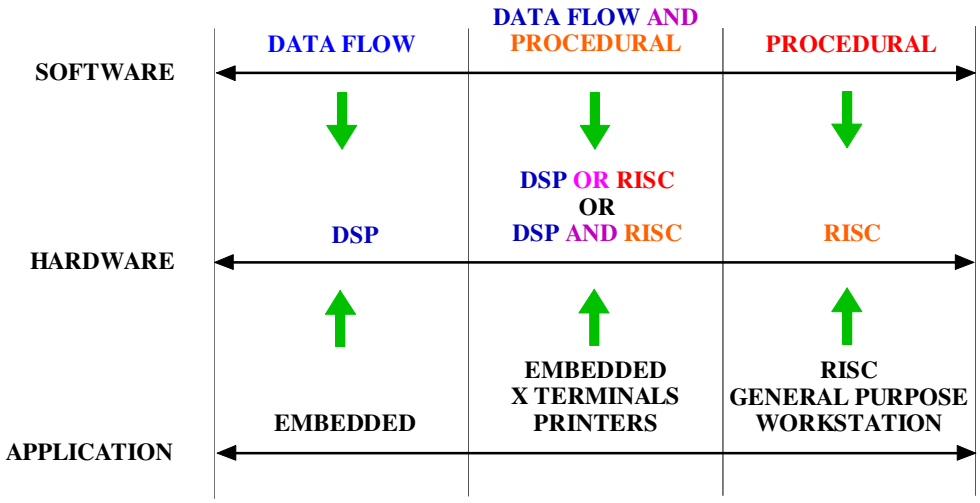
Gyula Fehér feher@novserv.obuda.kando.hu



Architectural support of protection

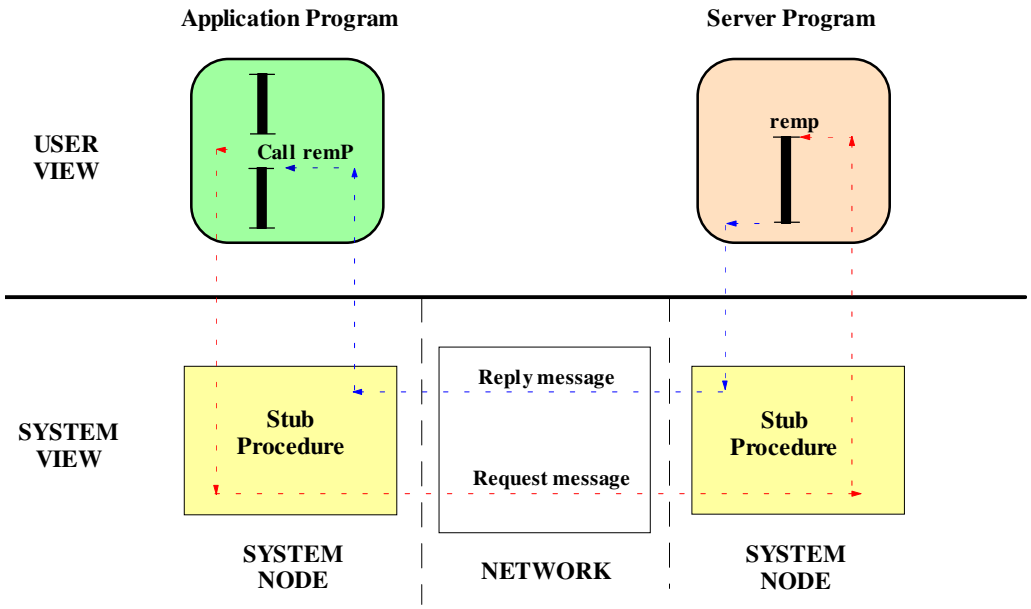
- **Effective means to protect system software from inadvertent user misshandling.**
- **Each process's programs and data must be protected, isolated so that another users's program is unable to "write" over them.**

Gyula Fehér feher@novserv.obuda.kando.hu



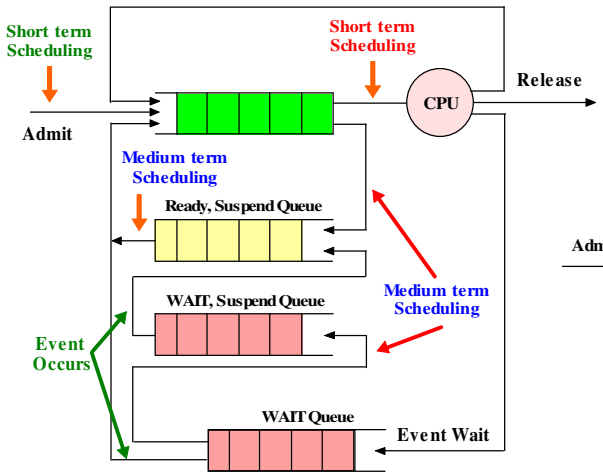
RISC vs. DSP

Gyula Fehér feher@novserv.obuda.kando.hu

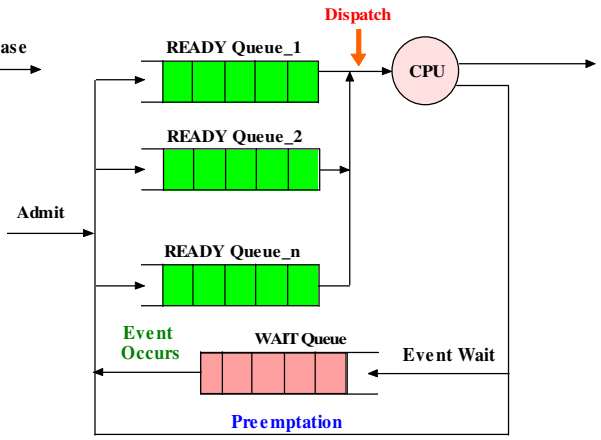


Remote Procedure Call

Gyula Fehér feher@novserv.obuda.kando.hu

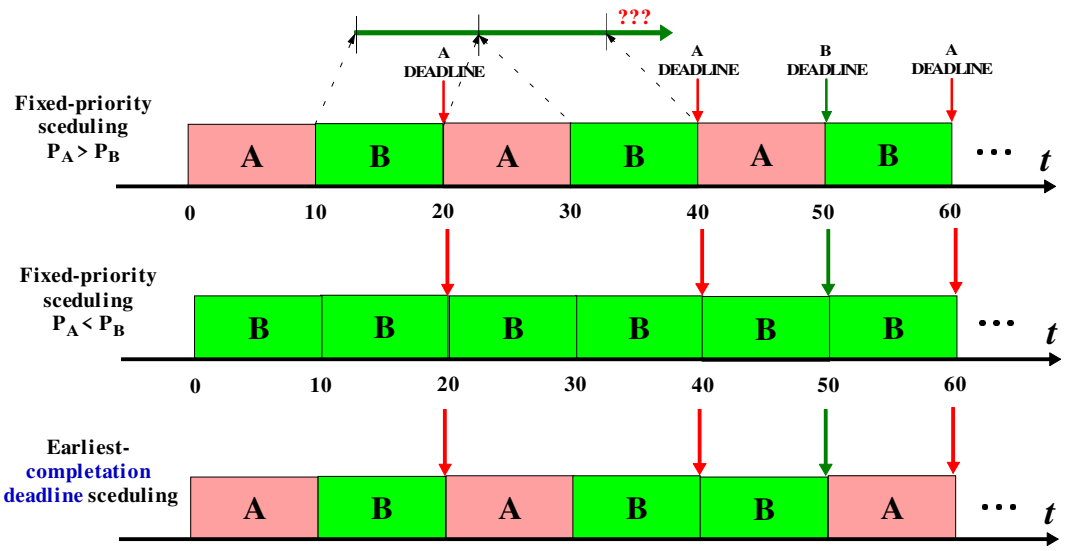


Queue diagram of Scheduling



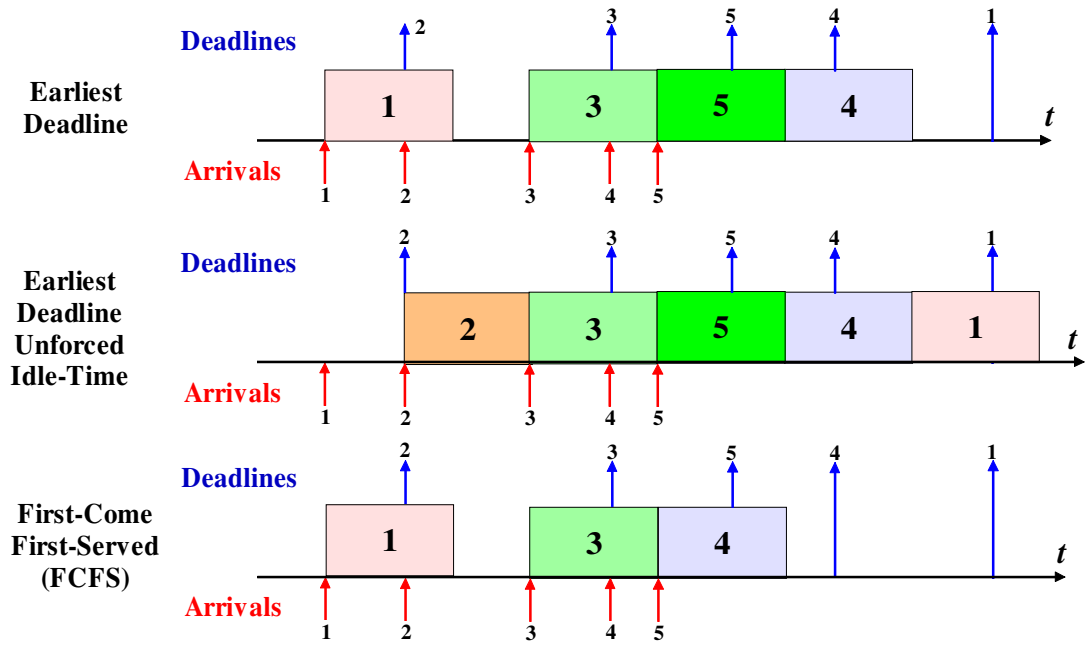
Priority Scheduling

Gyula Fehér feher@novserv.obuda.kando.hu



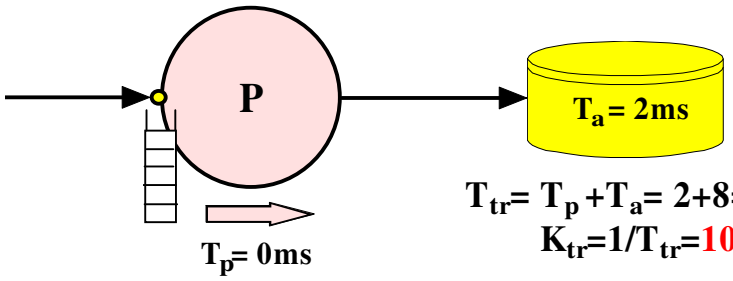
Scheduling of periodical real-time processes with completation deadlines

Gyula Fehér feher@novserv.obuda.kando.hu



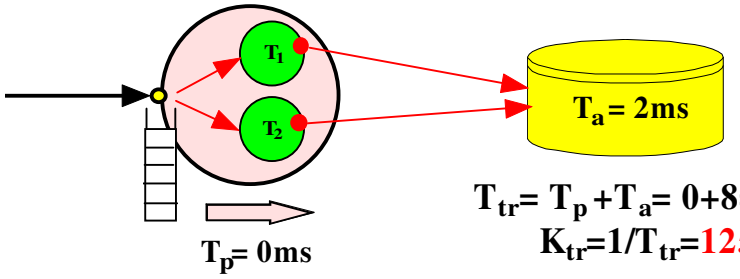
Sceduling aperiodic real-time processes with starting deadlines

Gyula Fehér feher@novserv.obuda.kando.hu



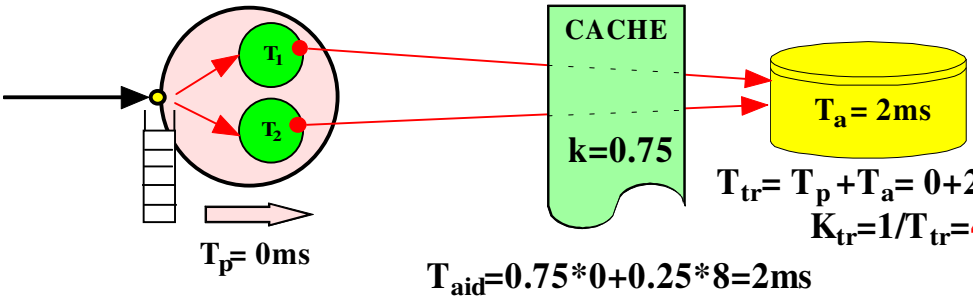
$T_{tr}= T_p + T_a = 2+8=10ms$

$K_{tr}=1/T_{tr}=100$



$T_{tr}= T_p + T_a = 0+8=8ms$

$K_{tr}=1/T_{tr}=125$



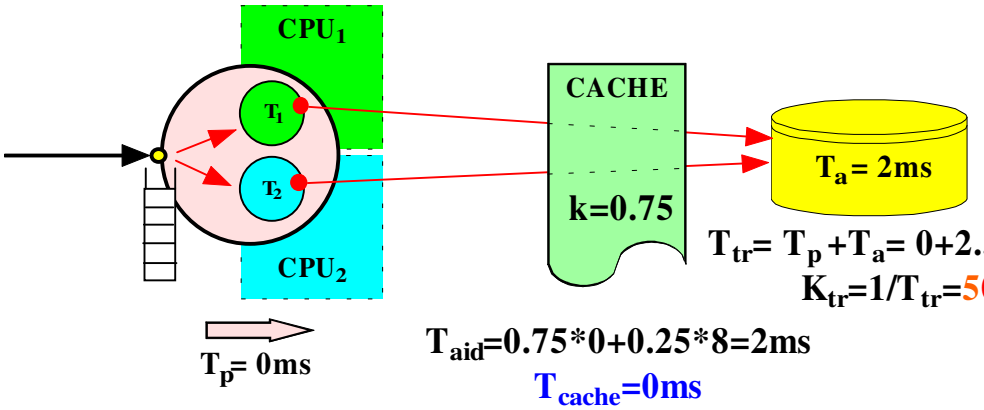
$T_{tr}= T_p + T_a = 0+2.5=2.5ms$

$K_{tr}=1/T_{tr}=400$

$T_{aid}=0.75*0+0.25*8=2ms$

$T_{cache}=0.5ms$

$T_a=T_{aid}+T_{cache}=2.5ms$



$T_{tr}= T_p + T_a = 0+2.5=2.5ms$

$K_{tr}=1/T_{tr}=500$

$T_{aid}=0.75*0+0.25*8=2ms$

$T_{cache}=0ms$

$T_a=T_{aid}+T_{cache}=2ms$