
# Memphis Manycore

**Apresentação** Angelo Elias Dalzotto
**Coordenador**: Prof. Dr. Fernando G. Moraes
**Adaptado da apresentação de** Dr. Marcelo Ruaro

PUCRS

GAPH
Grupo de Apoio ao Projeto de Hardware

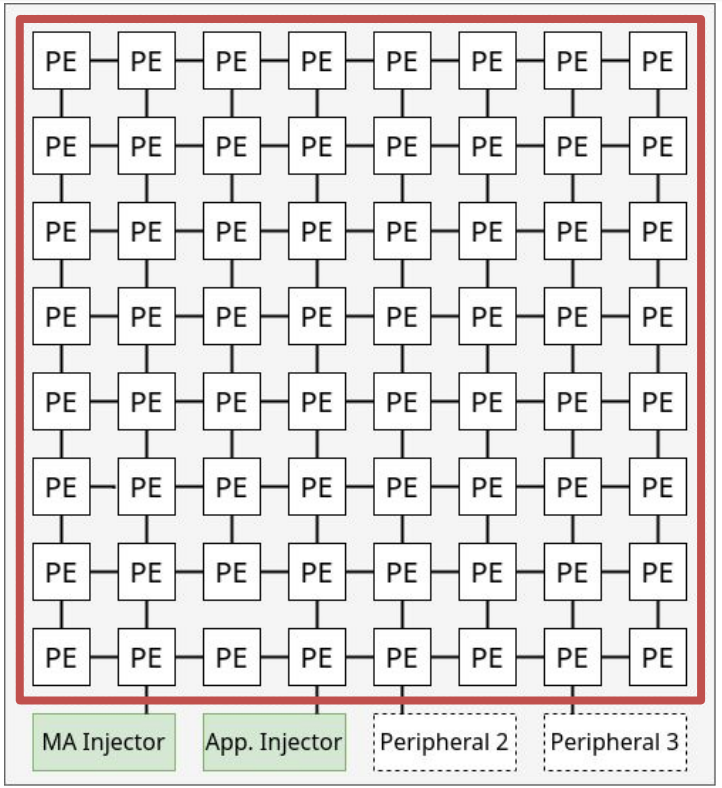
# Overview

# Memphis
## Many-core Modeling Platform for Heterogeneous SoCs



**GPPC** – **Homogeneous** region

**G**eneral **P**urpose **P**rocessing **C**ore

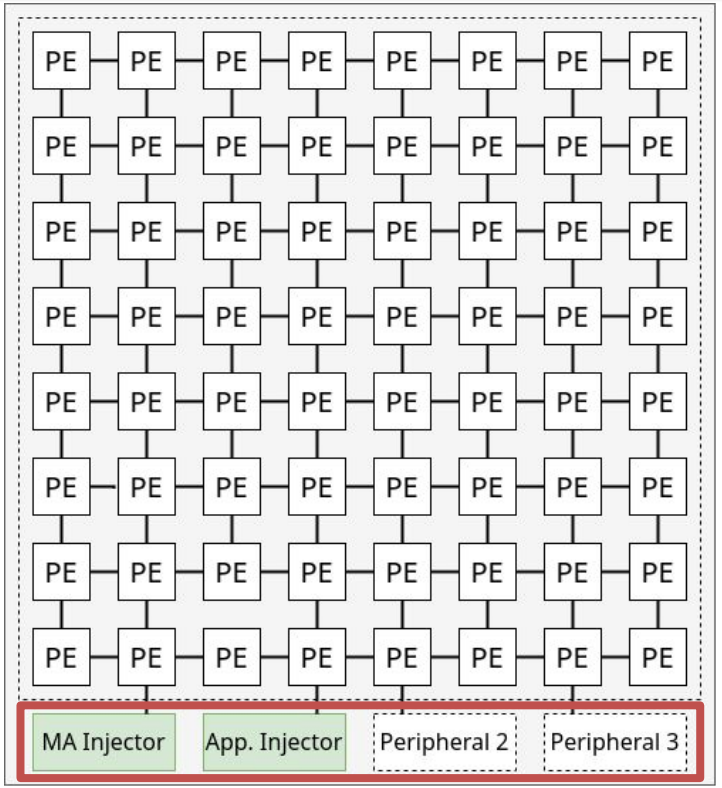


# Memphis
# Many-core Modeling Platform for Heterogeneous SoCs



## Peripherals – Heterogeneous region

- Connected to **PE** borders
- Provides:
  - I/O Interface
    - Default: **Application Injector**
    - Default: **MA Injector**
  - Hardware acceleration



# Hardware

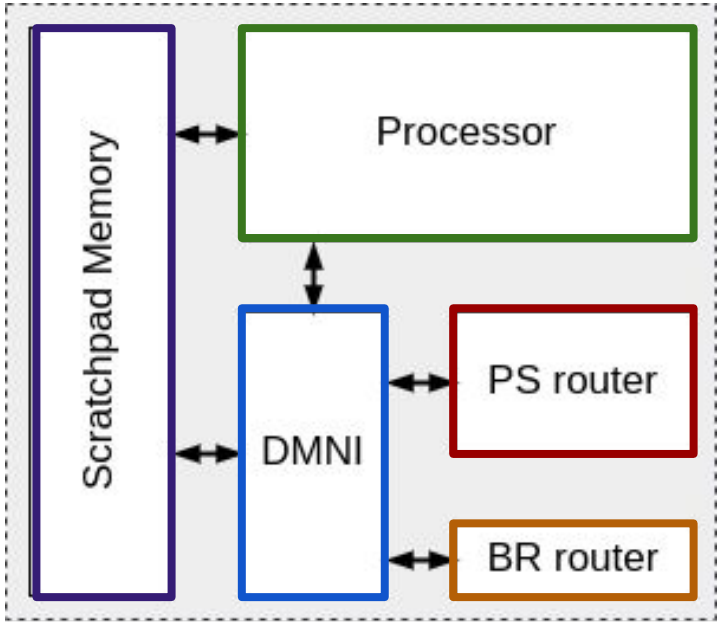
# Memphis[1]
# Many-core Modeling Platform for Heterogeneous SoCs



**PE** – **Homogeneous** region

**P**rocessing **E**lement

- *RISC-V* **CPU**
- ***DMNI*** (DMA + NI)
- *Scratchpad* **memory**
- *Hermes* **router**
- *BrNoC* **router**



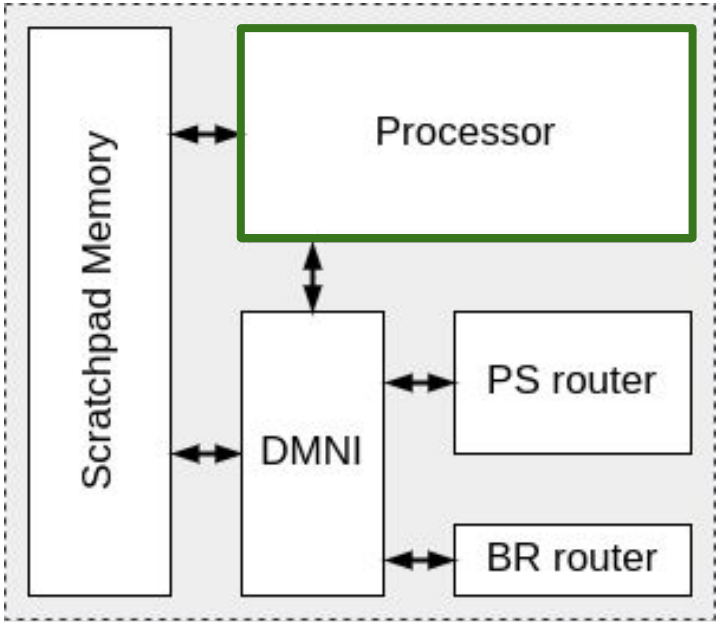

1. Ruaro, M., Caimi, L. L., Fochi, V., and Moraes, F. G. (2019). Memphis: a framework for heterogeneous many-core SoCs generation and validation. Design Automation for Embedded Systems, 23(3-4):103-122.

# Processor



## RISC-V CPU

- RV32IM *Instruction Set Architecture*
  - 32-bit
  - 32 registers
  - No floating-point
- Currently ISS-only
- Simple paging support

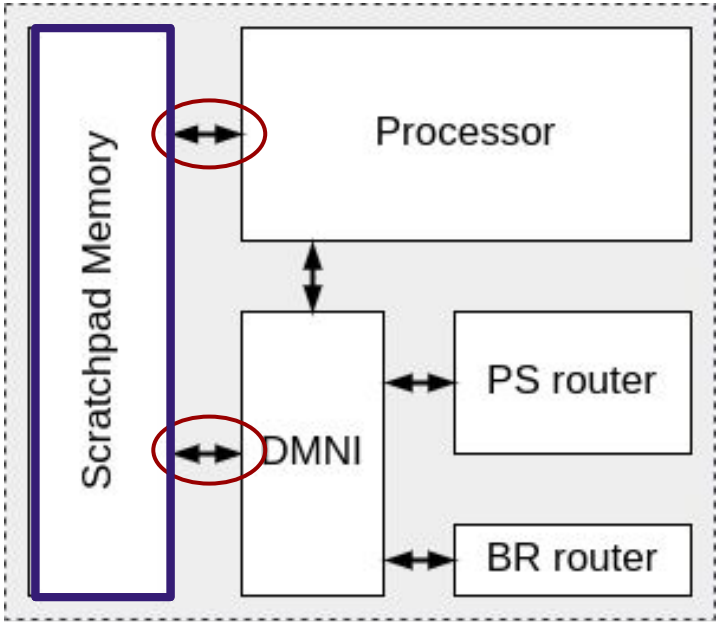


# Local Memory



## **Scratchpad** memory

- Dual-port
  - CPU access
  - DMNI access

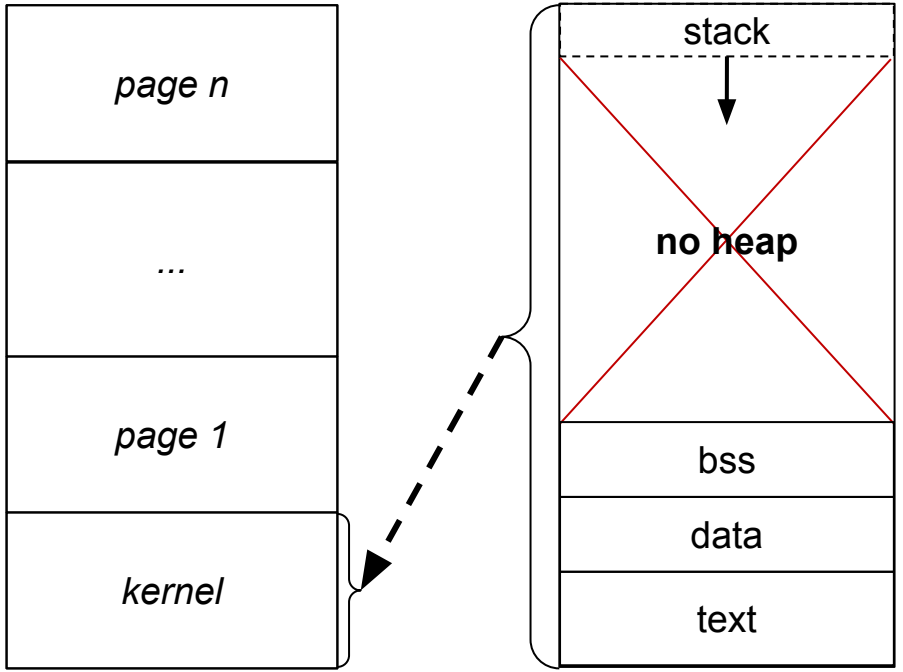


# Local Memory



## **Scratchpad** memory

- Dual-port
  - CPU access
  - DMNI access
- Parameterizable pages
  - Equal sizes
  - Currently no dynamic memory

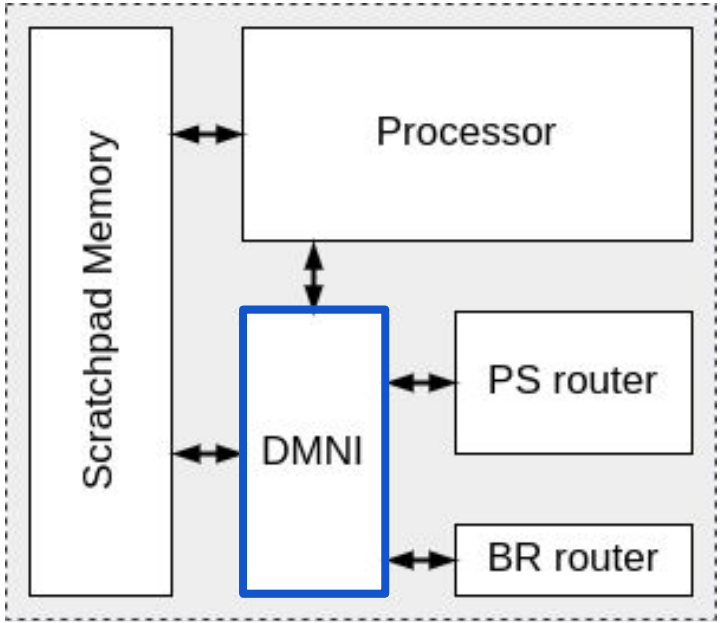


# DMNI



## Direct Memory Network Interface[2]

- Interface between local memory and NoC
- Specialized for NoC-based manycores




2. Ruaro, M., Lazzarotto, F. B., Marcon, C. A., and Moraes, F. G. (2016). DMNI: A specialized network interface for NoC-based MPSoCs. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), pages 1202-1205.

# Packet-Switching Router

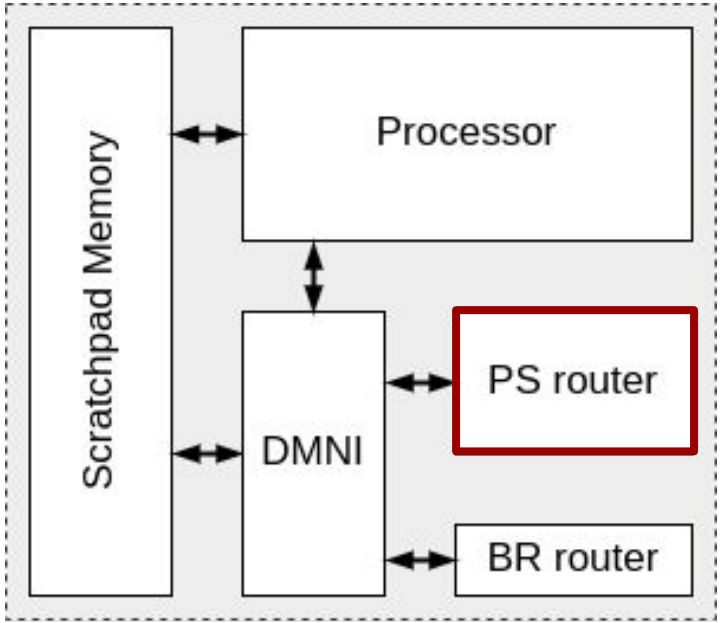


## Hermes Router[3]

- XY addressing and routing
- Packet switching
- Wormhole credit-based control flow

Hermes has other derivations:

- Asynchronous
- Virtual-channel
- Circuit-switching
- ...



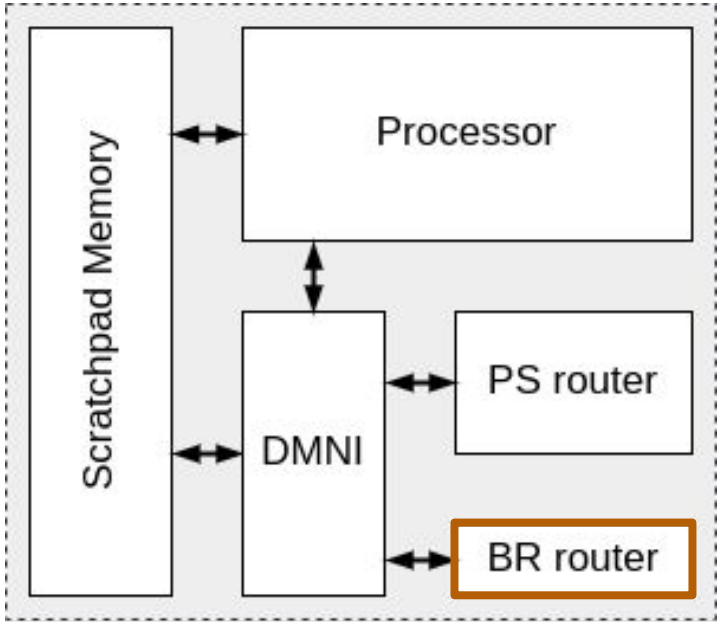

3. Moraes, F., Calazans, N., Mello, A., Möller, L., and Ost, L. (2004). HERMES: an infrastructure for low area overhead packet-switching networks on chip. Integration, 38(1):69-93.

# Broadcast Router



## BrNoC Router[4]

- Broadcast transmission only
  - Interrupts ALL or a TARGET PE
- Single-flit messages



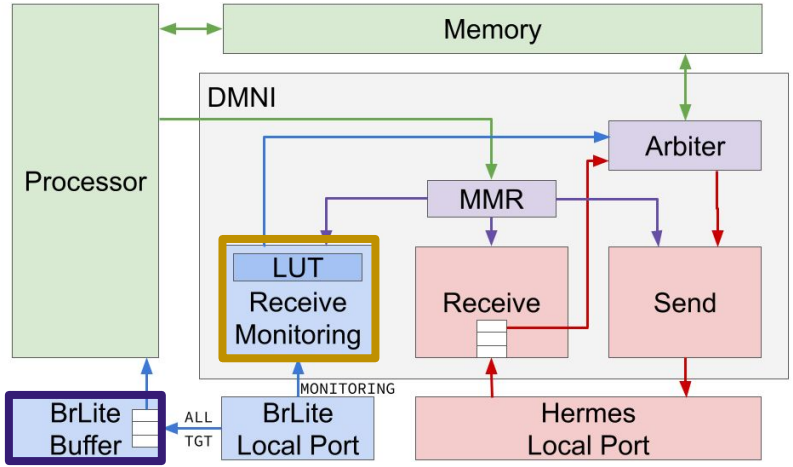

4. Wachter, E., Caimi, L., Fochi, V., Munhoz, D., Moraes, F. (2017). BrNoC: A broadcast NoC for control messages in many-core systems. Microelectronics Journal, 68:69-77.

# Broadcast Router



## BrNoC Router[4]

- Broadcast transmission only
  - Interrupts ALL or a TARGET PE
- Single-flit messages
- Buffered output
  - Transmission is faster than processing
- Monitoring messages through DMNI



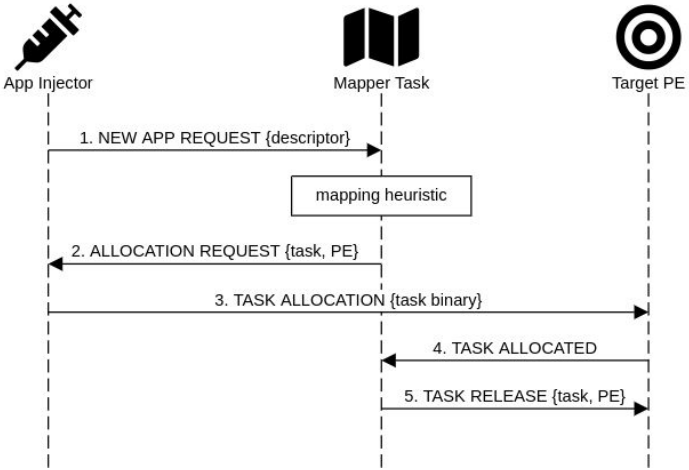

4. Wachter, E., Caimi, L., Fochi, V., Munhoz, D., Moraes, F. (2017). BrNoC: A broadcast NoC for control messages in many-core systems. Microelectronics Journal, 68:69-77.

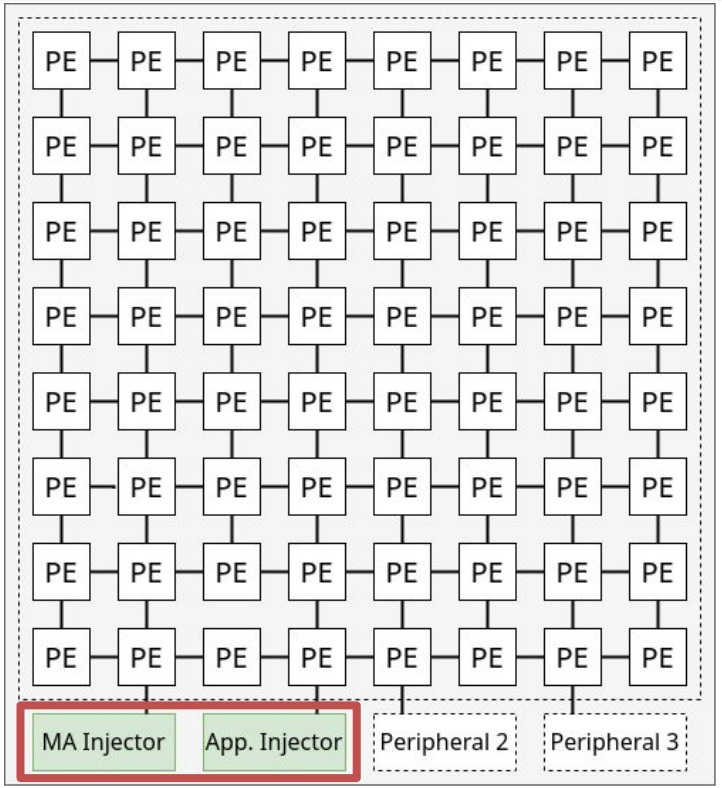
# Application Injector



## Injectors

- Interacts with external off-chip memory
- Loads applications into the manycore
- Separated injection for secure content (MA Inj



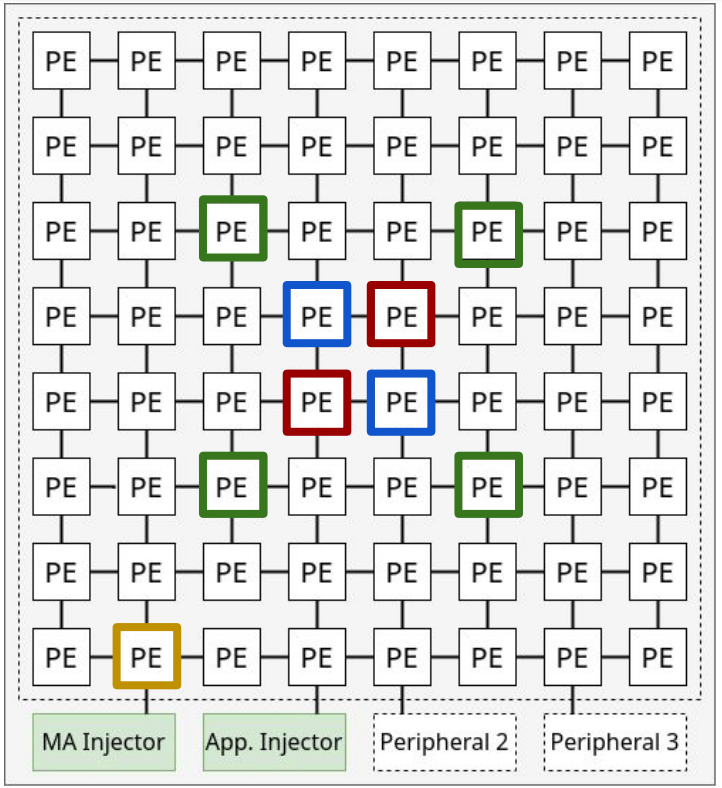
# Software

# Management Application[5]



- Division of management tasks
- **Observer tasks**
- **Decider tasks**
- **Actuator tasks**

- **Mapper task**[6]




5. Ruaro, M., Santana, A., Jantsch, A., and Moraes, F. G. (2021). Modular and Distributed Management of Manycore SoCs. ACM Transactions on Computer Systems (TOCS), 38(1-2):1-16.

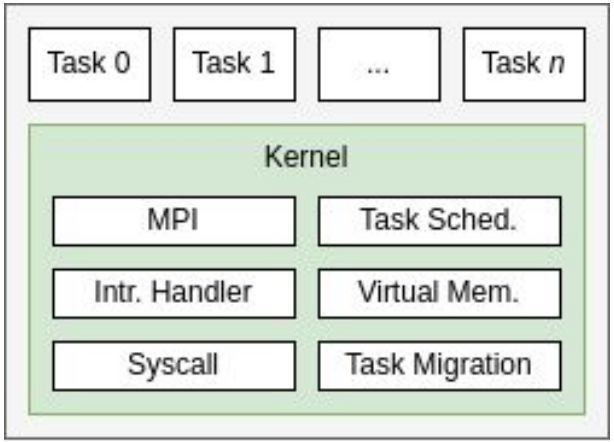
6. Dalzotto, A. E., Ruaro, M., Erthal, L. V., and Moraes, F. G. (2021). Dynamic Mapping for Many-cores using Management Application Organization. In Proceedings of the International Conference on Electronics, Circuits, and Systems (ICECS), pages 1-6.

# Operating System



**Kernel**

- User tasks
  - Virtual Memory
  - Scheduler
- System Calls
- Message Passing Interface
- Actuation Mechanisms
  - Task migration support



# Scheduler



**Best effort**

- Round-robin

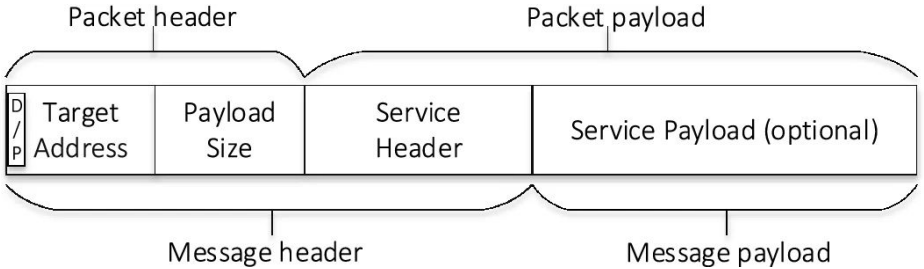
**Real-time**[7]

- LST
  **L**east **S**lack **T**ime

7. Ruaro, M., and Moraes, F. G. (2016). Dynamic real-time scheduler for large-scale MPSoCs. In Proceedings of the Great Lakes Symposium on VLSI, pages 341-346.

# NoC packet and message structure






**Packet Header** - seen by NoC:

- Target PE
- Payload Size

**Packet Payload** - seen by kernel:

- Service
- 10 configurable flits
- Optional payload with variable size

# BrNoC message structure




**Header - NoC view only**:

- ID - retransmission control
- SVC - ALL/TARGET/CLEAR
- SRC and TGT
- PROD - producer task ID

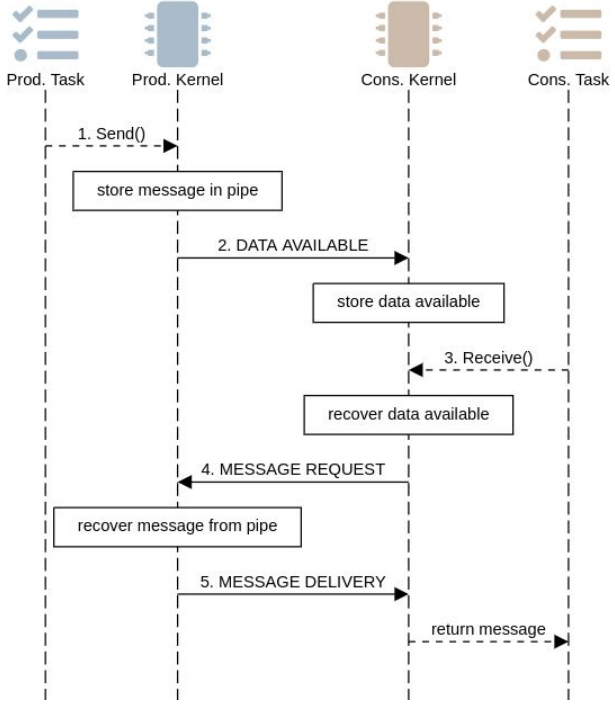
**Payload** - seen by kernel:

- Additional 40-bit payload
  - 8-bit system service
  - 32-bit actual payload

# Inter-task Communication



Tasks communicate using **Send** and **Receive** primitives

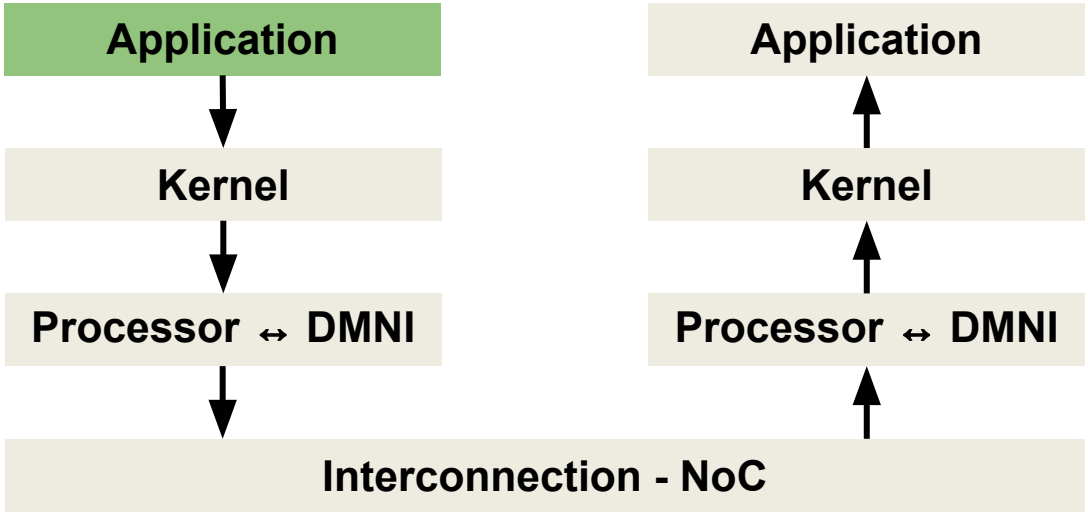


# Communication Layers



## Application

- Sends the message by calling the **Send** API primitive

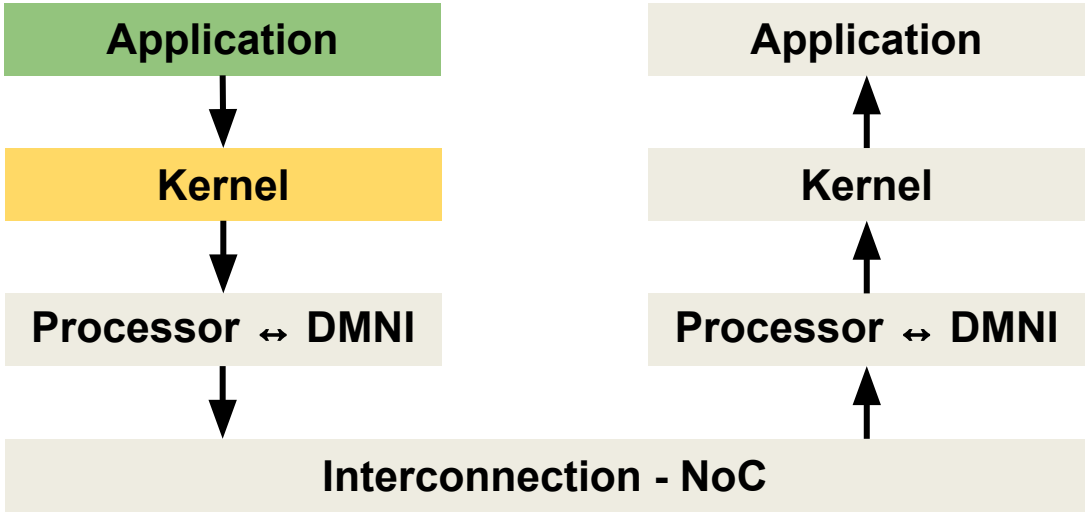


# Communication Layers



## Kernel

- Stores the message in the pipe and creates a packet
- Configures the DMNI to send the message

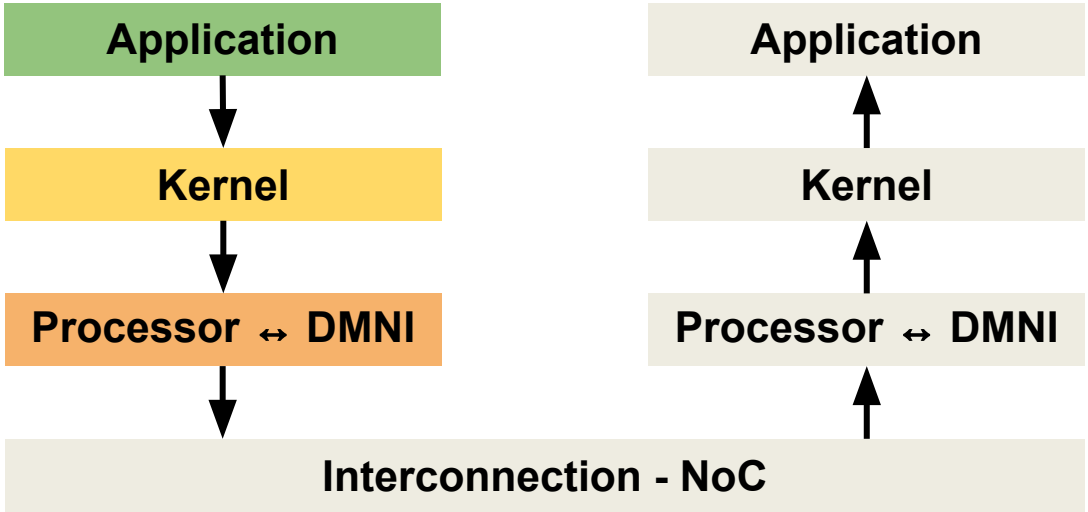


# Communication Layers



## DMNI

- Copies packet and pipe to NoC
- Serialization and flow control

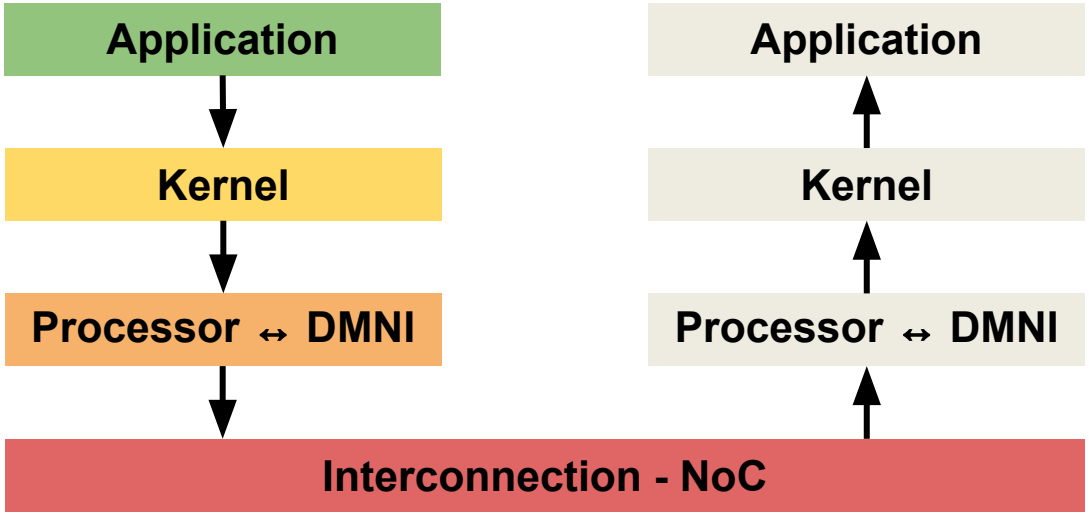


# Communication Layers



## NoC

- Sends the message to the target PE
- Divided in 32-bit flits

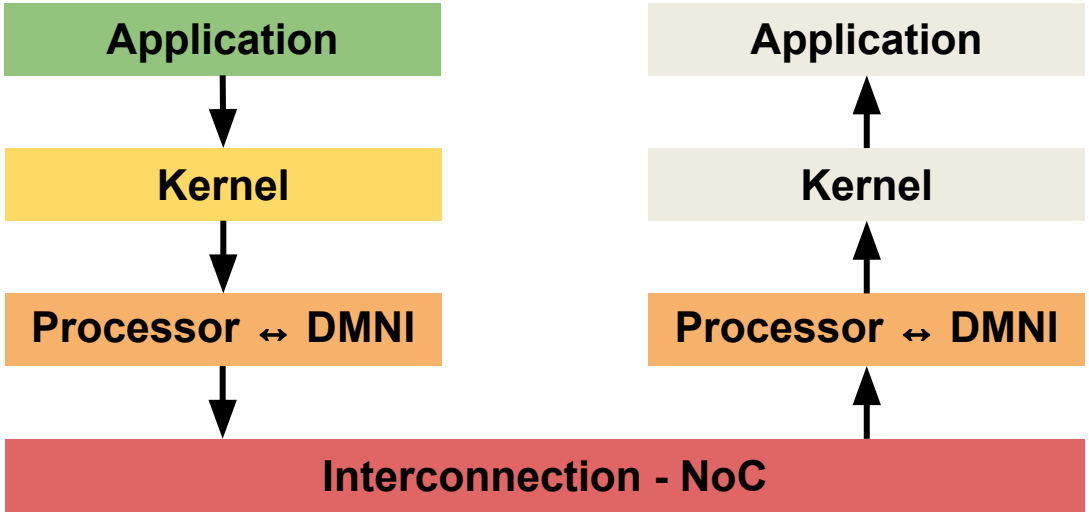


# Communication Layers



## DMNI

- Receives packet containing MESSAGE_DELIVERY
- Deserialization and flow control

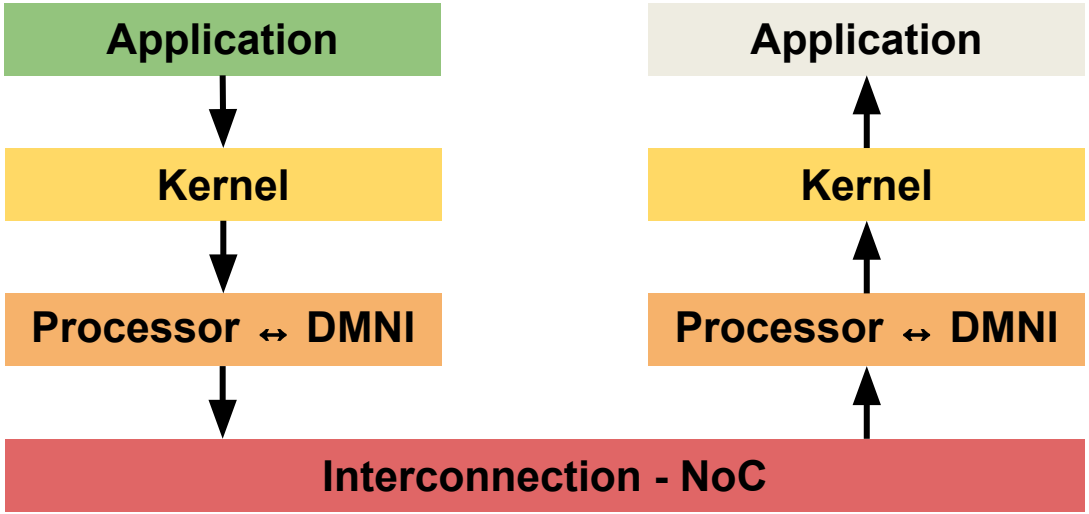


# Communication Layers



## Kernel

- Handles the packet
- Programs the DMNI to receive the message

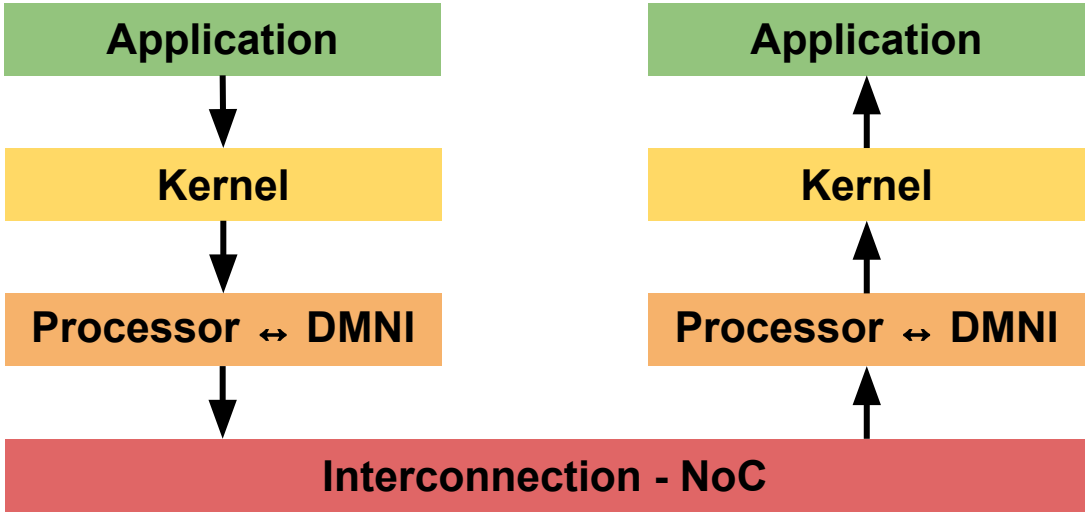


# Communication Layers



## Application

- The execution is returned to the application
- Message is stored in the pointer set by the **Receive** syscall



# BrNoC Services

| 39 38 37 36 35 34 33 32 | 31 30 29 28 27 | 26 25 24 | 23 22 21 20 19 18 17 16 | 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|---|---|
| RELEASE_PERIPHERAL | – | – | – | – |
| CLEAR_LUT | – | – | – | TASK_ID |
| ANNOUNCE_MONITOR | – | MON_TYPE | – | TASK_ID |
| DATA_AV | – | – | PROD_ADDR | CONS_ID |
| MESSAGE_REQUEST | – | – | CONS_ADDR | PROD_ID |
| TASK_MIGRATION | – | – | ADDR | TASK_ID |
| MONITORING | MON_PAYLOAD | | | |

# Hermes Services

| | |
|---|---|
| DATA_AV (to/from periph.) | Avoid broadcast DoS |
| MESSAGE_REQUEST (to/from periph.) | |
| MESSAGE_DELIVERY | Deliver message with variable-size payload |
| MIGRATION_* | |

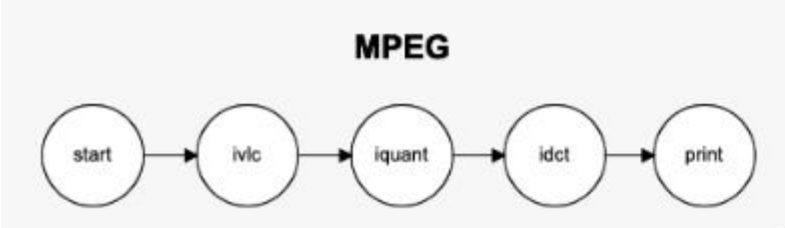
# Applications

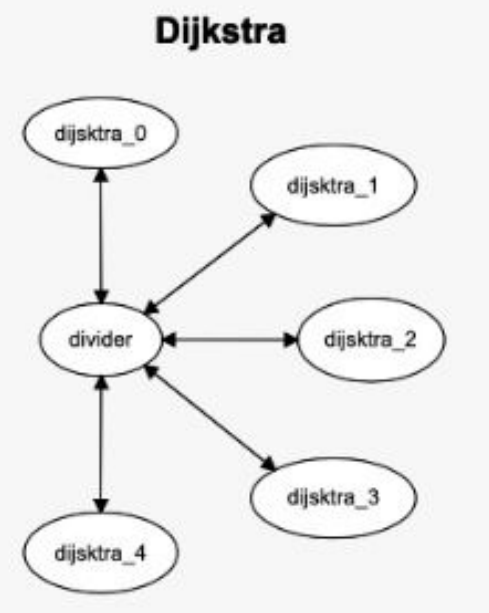
# Application



**Application**

- Set of **tasks**
- Modeled as a CTG (**C**ommunicating **T**ask **G**raph)



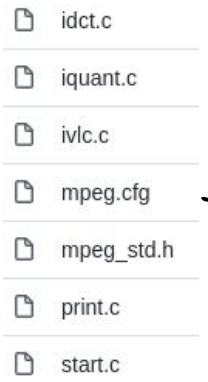



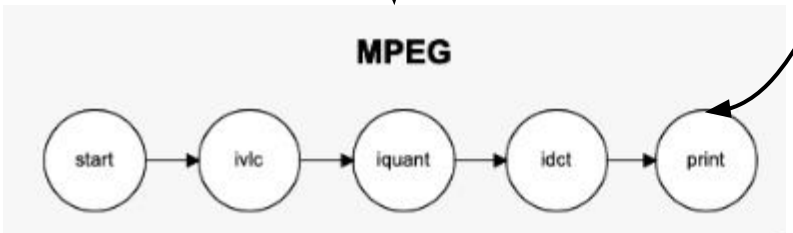
# Task



## Task

- Performs computation and communication
  - Each .c file is a task




```c
int main()
{
    printf("MPEG frames:\n");

    for(int i = 0; i < MPEG_FRAMES; i++)
    {
        memphis_receive(&msg1, idct);
        printf("%d\n", memphis_get_tick());
    }

    printf("End Task PRINT - MPEG\n");

    return 0;
}
```

# #include <memphis.h>



```
typedef struct {
        int length;
        int msg[MSG_SIZE];
} message_t;
void memphis_send(message_t *msg, unsigned cons_id);
void memphis_receive(message_t *msg, unsigned prod_id);
void memphis_send_any(message_t *msg, unsigned cons_id);
void memphis_receive_any(message_t *msg);
void memphis_br_send_all(unsigned payload, char service);
void memphis_br_send_tgt(unsigned payload, unsigned cons_id, char service);
unsigned memphis_get_tick();
unsigned memphis_get_id();
unsigned memphis_get_addr();
void memphis_real_time(unsigned period, unsigned deadline, unsigned exec_time);
```

# Generation[8]

8. Castilhos, G., Wachter, E., Madalozzo, G., Erichsen, A., Monteiro, T., and Moraes, F. (2014). A framework for mpsoc generation and distributed applications evaluation. In Proceedings of the International Symposium on Quality Electronic Design (ISQED), pages 408-411.

# Testcase



## Features of the platform

File: example_testcase.yaml

```
sw:
  max_tasks_app: 10
hw:
  page_size_KB: 32
  stack_size: 1024
  tasks_per_PE: 4
  mpsoc_dimension: [3,3]
  Peripherals:
    - name: APP_INJECTOR
      pe: 2,2
      port: N
    - name: MAINJECTOR
      pe: 0,0
      port: S
```

$ memphis testcase example_testcase.yaml

- Creates folders and copies files
- Builds hardware and kernel

# Scenario



## Applications to evaluate

File: example_scenario.yaml

```
management:
 - task: mapper_task
   static_mapping: [0,0]
 - task: migration_task
   static_mapping: [0,1]
 - task: rt_monitor
   static_mapping: [0,2]
apps:
 - name: synthetic
 - name: prod_cons
   start_time_ms: 5
   static_mapping:
     prod: [1,1]
```

$ memphis scenario example_testcase/ example_scenario.yaml

- Copies applications
- Builds applications
- Generates repository (MA/Application Injector memory)

# Debugging

# Debugging Framework[9]



**Data Extraction** (back-end)

- Extracts simulated data from platform
- Inserts into logs

**Graphical Debugging** (front-end)

- Read extracted data from logs
- Enable easy debuggability by the graphical features

9. Ruaro, M., Carara, E. A., and Moraes, F. G. (2014). Tool-set for NoC-based MPSoC debugging—A protocol view perspective. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), pages 2531-2534.

# Main View



**Debug possibilities**:

- Communication flows
- Routing
- Link utilization
- Management Protocols
- Parallel communications

# Mapping View



## Debug possibilities:

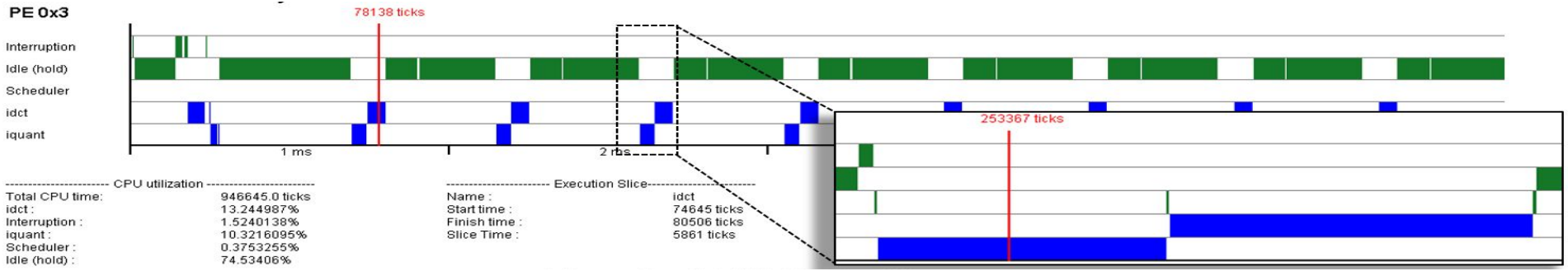
- Task mapping algorithm
- PEs occupation
- Task execution status

# CPU Utilization View



## Debug possibilities:

- Scheduling algorithms
- OS and task bugs
- Other software malfunctions



# Resources

# Resources



**Code, installation and usage instructions:**

https://github.com/gaph-pucrs/MA-Memphis

**Debugger tutorial (pt):**

https://youtu.be/nvgtvFcCc60