# Practical Work 3
# DAI - HEIG-VD

Loïc Herman Massimo Stefani

## I. Multicast Distributed System Monitoring Protocol

### A. Overview

The Multicast Distributed System Monitoring Protocol (MDSMP) serves as a crucial tool for efficiently transmitting and collecting system metrics in distributed environments. Designed to streamline the process of monitoring, MDSMP enables systems to send metrics data to a central monitoring server. This protocol is especially well-suited for scenarios where real-time insights into system performance are essential.

### B. Protocol

MDSMP employs text-based messages following the Prometheus metrics format, as described in the Prometheus documentation (https://prometheus.io/docs/concepts/data_model/). This standardization ensures compatibility with Prometheus-based monitoring and facilitates easy integration into existing monitoring ecosystems.
Since MDSMP is based on UDP, it operates without establishing a formal connection between clients and the server. This results in a lightweight and efficient communication model, reducing overhead and latency.
The port used for all multicast messages is 9378.
Multicast addresses groups are defined as follows:

- cpu : 230.0.0.1
- ram : 230.0.0.2
- dsk : 230.0.0.3

All messages exchanged within the MDSMP protocol must be encoded in UTF-8. This standardized encoding scheme ensures compatibility across diverse systems and prevents data corruption during transmission.

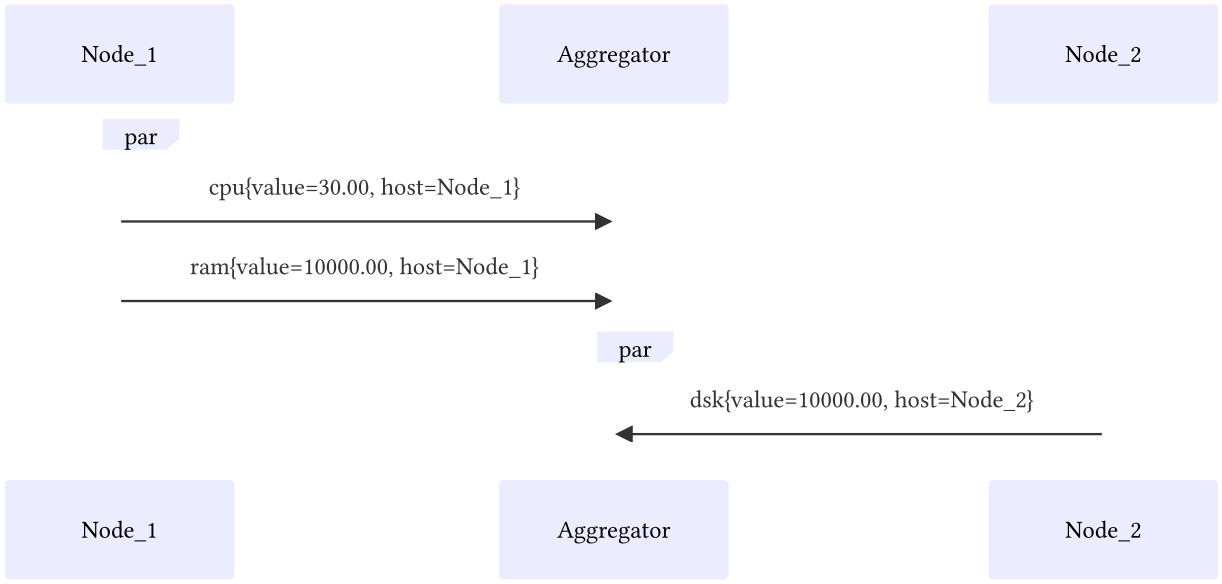
### C. Messages

1) *Sending data:*

1. `<type>{value=<value>, host=<host>}`
    1. `<type>` : Type of the data sent (cpu, ram, dsk)
    2. `<value>` : Value of the sent data
        1. cpu : CPU consumption in percentage
        2. ram : RAM consumption in MB
        3. dsk : Disk consumption in MB
    3. `<host>` : Name of the machine sending the data (hostname). Must be unique for each machine.

### D. Examples

The next figure contains an example of two nodes sending data to the aggregator. Since the messages are sent using the fire-and-forget method messages are sent without expecting anything in return, the messages can also be sent in parallel.


Node_1
Aggregator
Node_2
par
cpu{value=30.00, host=Node_1}
ram{value=10000.00, host=Node_1}
par
dsk{value=10000.00, host=Node_2}
Node_1
Aggregator
Node_2