ADiS-2010

CISIS International Conference on Complex, Intelligent and Software Intensive Systems

A Failure Detection System for Large Scale Distributed Systems

Andrei Lavinia, Ciprian Dobre, Florin Pop, Valentin Cristea

University POLITEHNICA of Bucharest
Romania

Outline

- Motivation
- Context
- Architecture
- Implementation
- Results
- Conclusions

Motivation

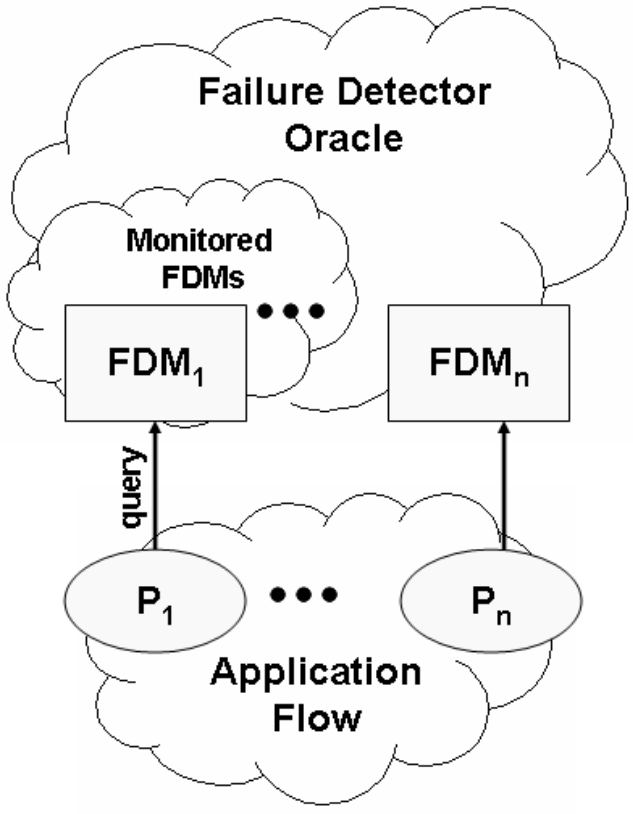
- Large scale distributed systems are hardly ever "perfect".
- Existing research projects in fault tolerance often offer only partial solutions.
- Fault Detection = a difficult problem
  - ➔ The geographical distribution of resources and users that implies frequent remote operations and data transfers.
  - ➔ The volatility of the resources, which are usually available only for limited periods of time.
  - ➔ The fault detector must be very sensitive to dynamic network conditions.
  - ➔ The different QoS restrictions coming from the applications being executed in such systems.

Context

- Problems with failure detectors:
  - ➔ Large number of messages
  - ➔ Scalability
  - ➔ Messages being lost
  - ➔ Flexibility
  - ➔ Dynamism
  - ➔ Security

Federation of unreliable failure detector modules

In distributed applications, failure detection is generally implemented through the use of directly invoked local services (unreliable local failure detectors).



FD schemas

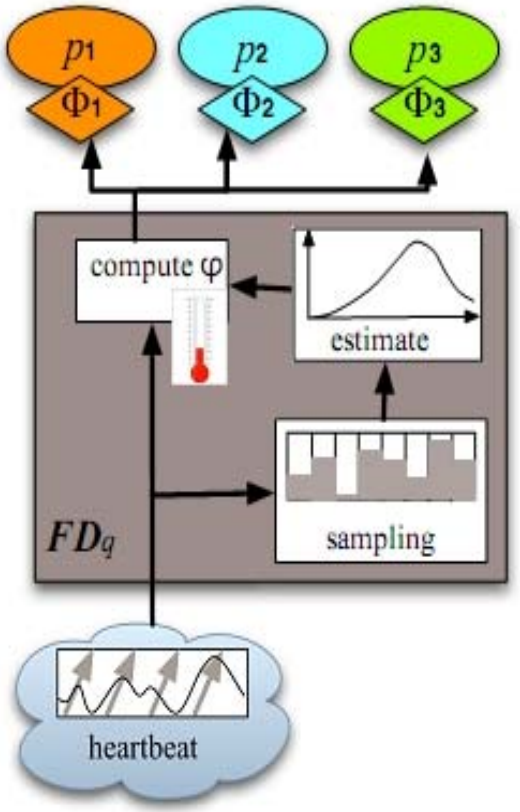
- Heartbeat strategy – the most common implementation
- Problem – scalability
  - ➔ Solution 1) Hierarchical FD structures – Two-level Globus FD
  - ➔ Solution 2) Gossip-like protocols - with high probability, eventually all processes obtain any piece of information
- Adaptive protocols
  - ➔ Adapt dynamically to changing network conditions
  - ➔ E.g.: A protocol that adjusts the timeout by using the maximum arrival interval of heartbeat messages
- Accrual failure detectors
  - ➔ Detector modules that associate to each of the monitored processes a real number value that changes over time
  - ➔ E.g.: The φ-failure detector samples the arrival time of heartbeats and maintains a sliding window of the most recent samples; the window is used to estimate the arrival time of the next heartbeat.

Adaptive FD protocols

- Protocols that adapt based on
  - ➔ Network conditions
  - ➔ Application requirements
- CHEN-FD and BERTIER-FD
  - ➔ Probabilistic analysis of network traffic and application requirements
  - ➔ Estimate next heartbeat arrival
- Disadvantages:
  - ➔ Accuracy related to the estimation function being used
  - ➔ To adapt to different application requirements the FD must manage multiple timeout values

Accrual detectors

- A monitored process is associated with a suspicion level
  - ➔ Increases when the process fails
- Applications interpret the suspicion level according to their own requirements



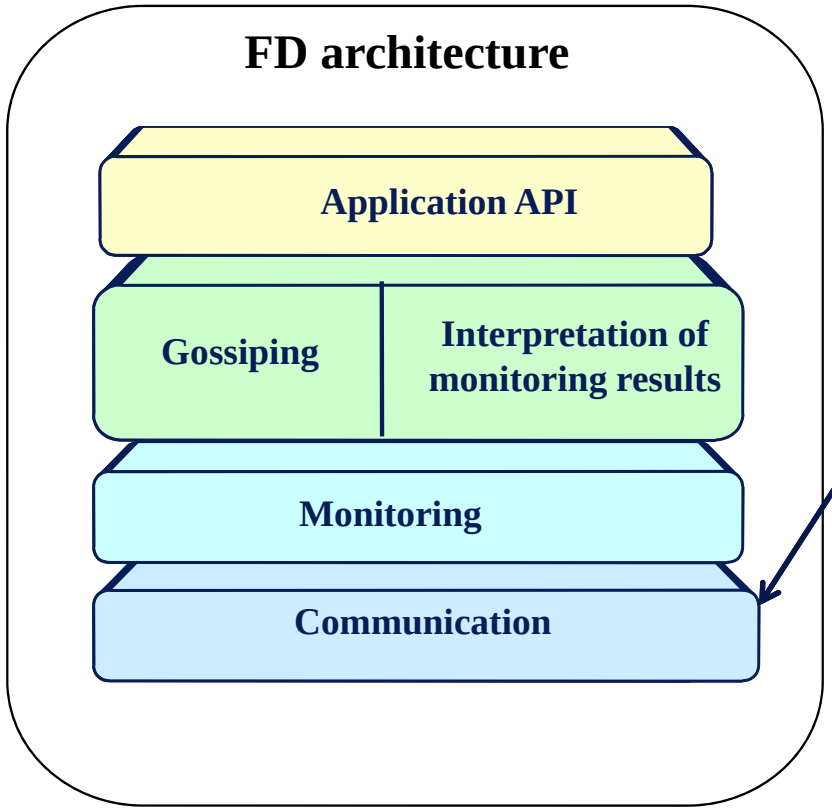
The approach

ADiS

Generic FD service for LSDS that:

- Adapts to changing network traffic conditions (adaptive FD) and different FT application requirements (accrual FD)
- Performs well in terms of scalability and dynamic nature of the system (clustering)
- Provides increases levels of trust when a process fails (gossiping)
- Decuples monitoring from interpretation
- Any application can (un)register for error notifications, can request the monitoring of a certain process, can interrogate the FD for the suspicion level associated with a certain process (application module)

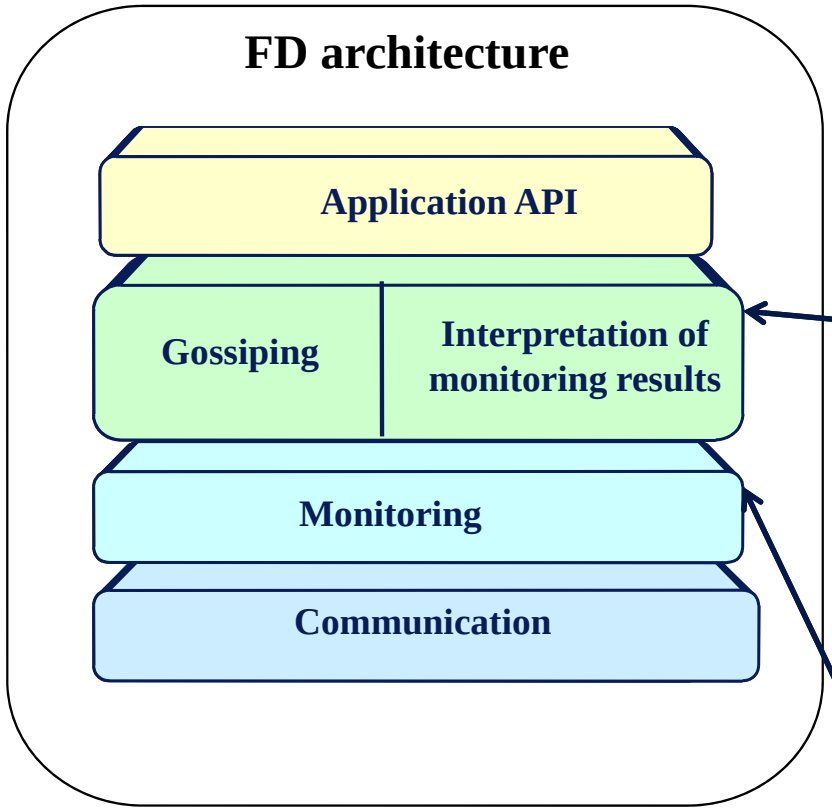
FD Architecture



Communication

- Hierarchical approach -> scalability
- Highly dynamic environment

FD Architecture



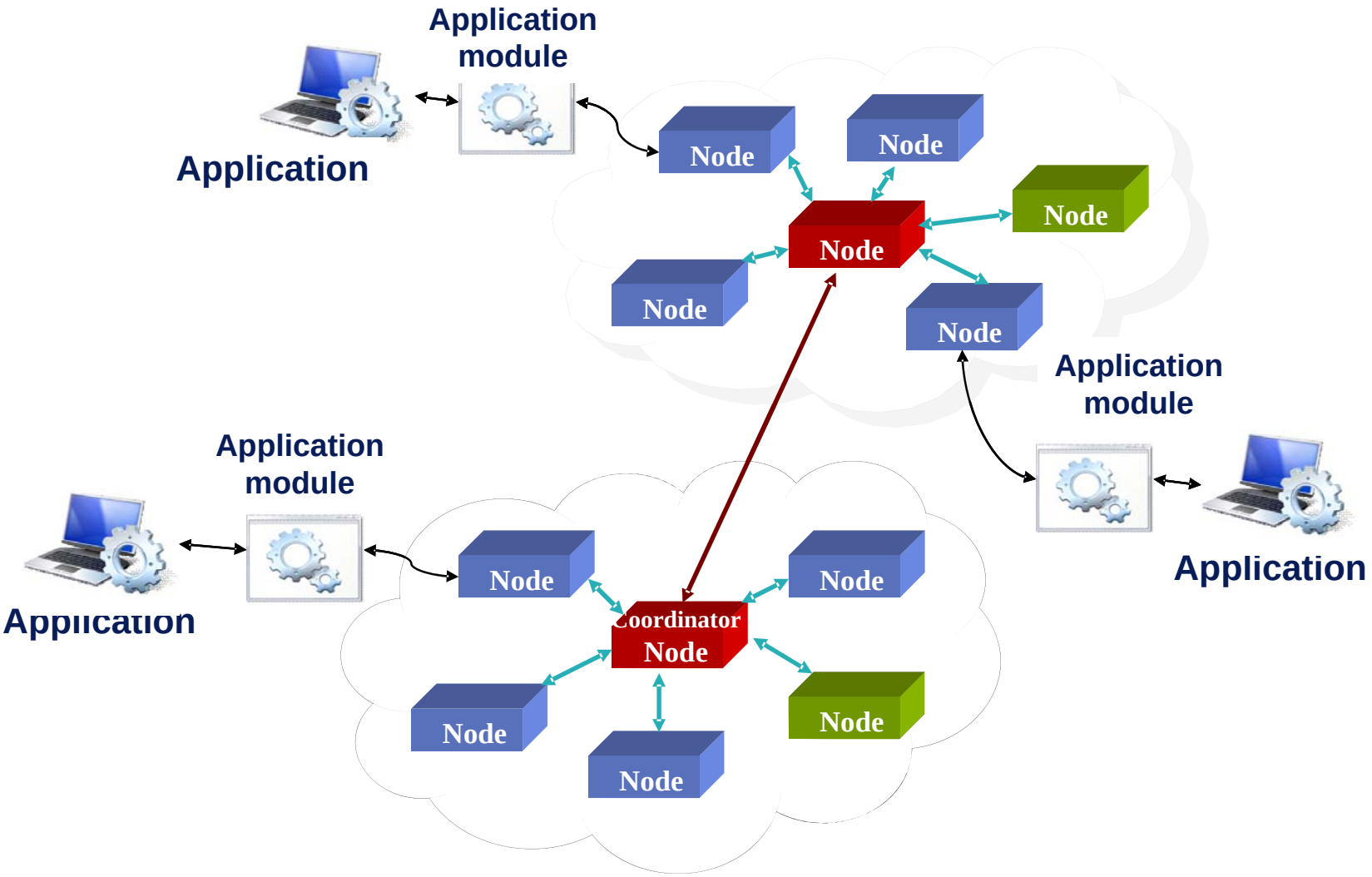
Interpretation

- Exponential Moving Average prediction method
- Adaptively adjusts the smoothing factor

Monitoring

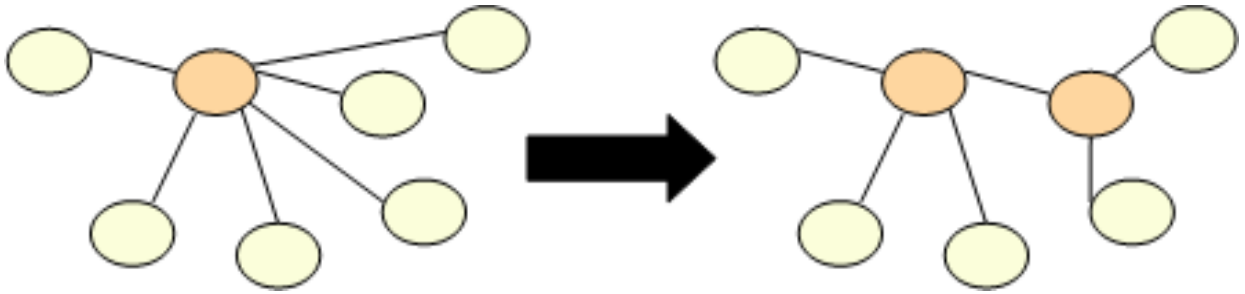
- Heartbeat messages

Hierarchical Communication

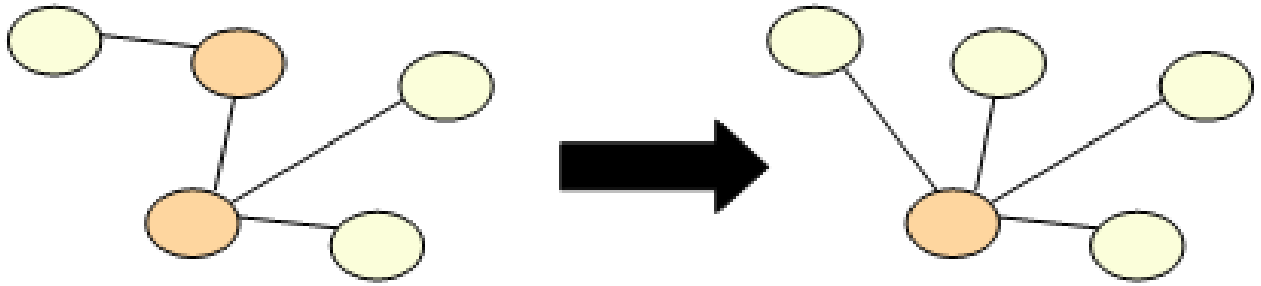


Clustering

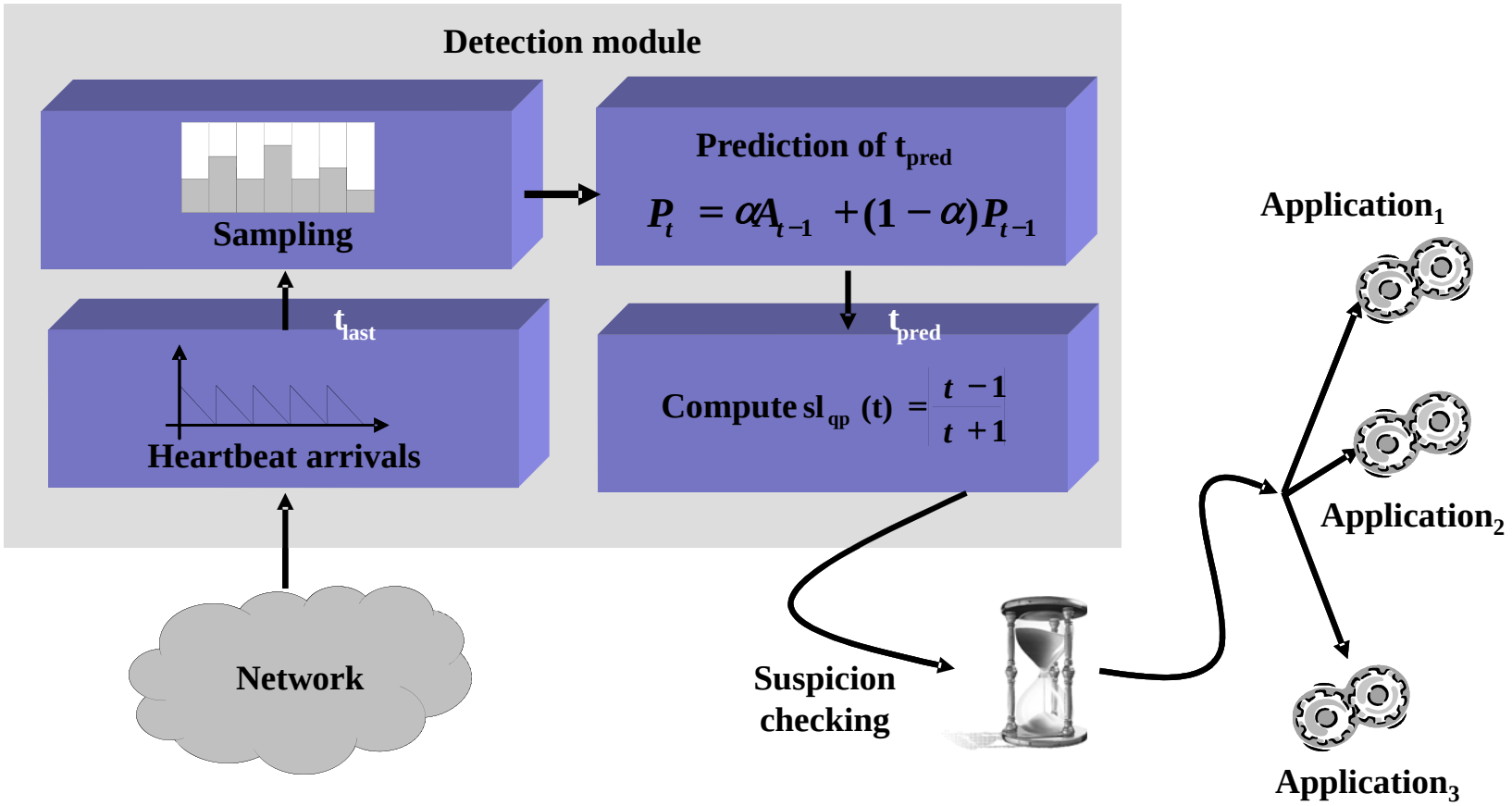
Creation of a new cluster



Merging of clusters



Interpreting monitoring data (1)



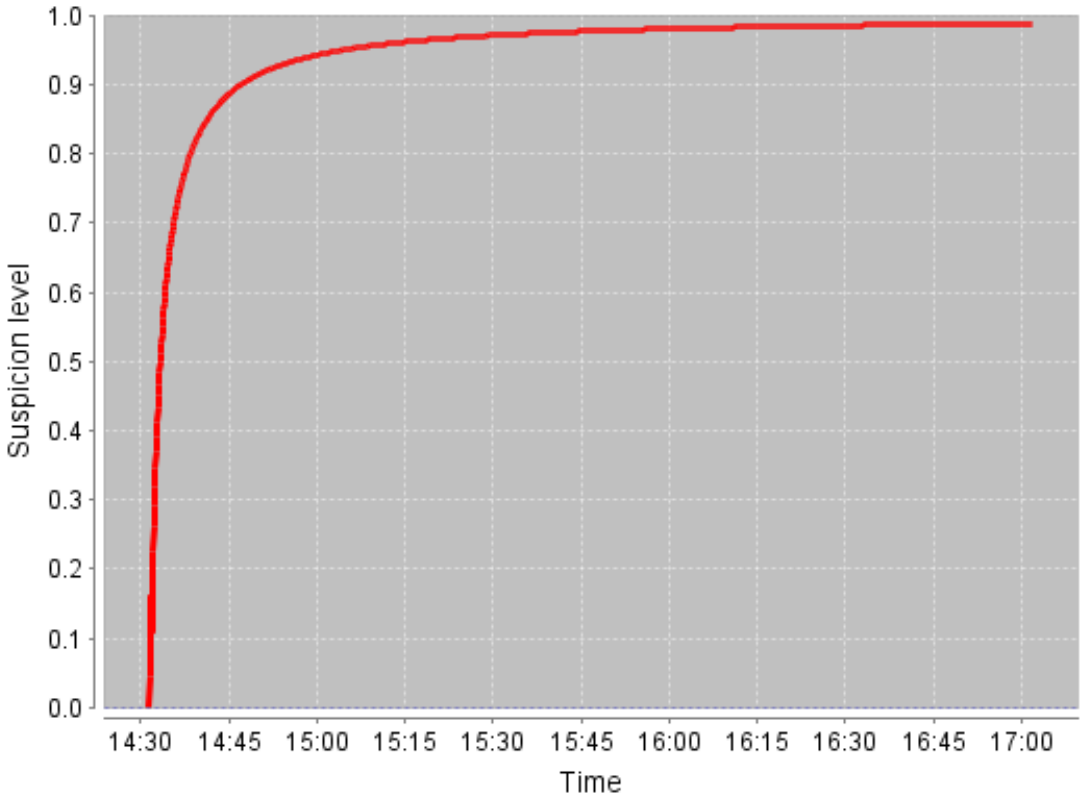
Interpreting monitoring data (2)

$$0 <= sl_{qp}(t) < 1 \mapsto sl_{qp}(t) = \frac{t-1}{t+1},$$

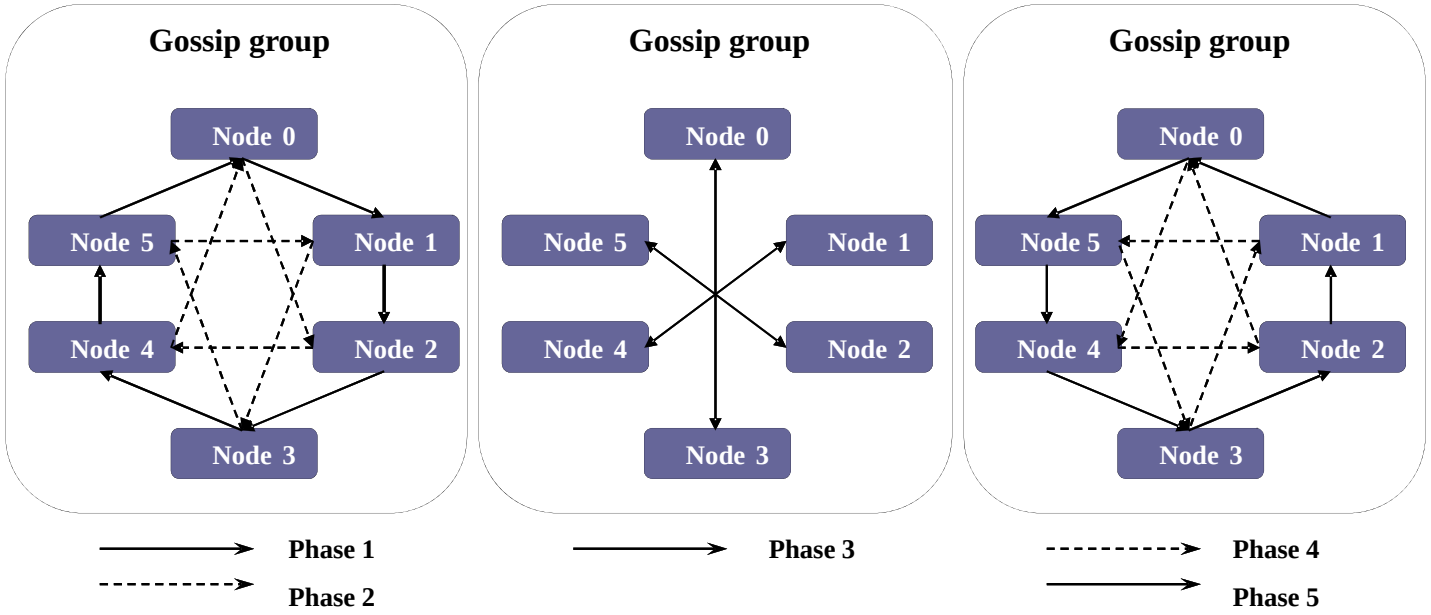
$$t = \frac{t_{now}}{t_{pred}} \in (0, \infty),$$

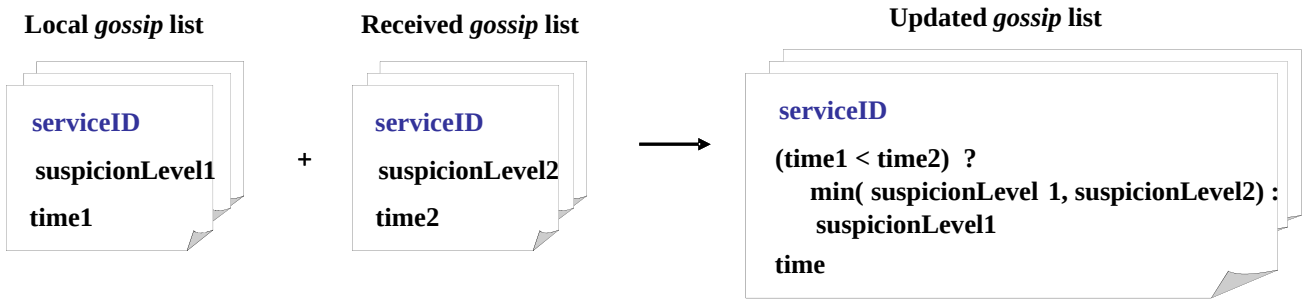
t_{now} *current value ,*

t_{pred} *last predicted value*

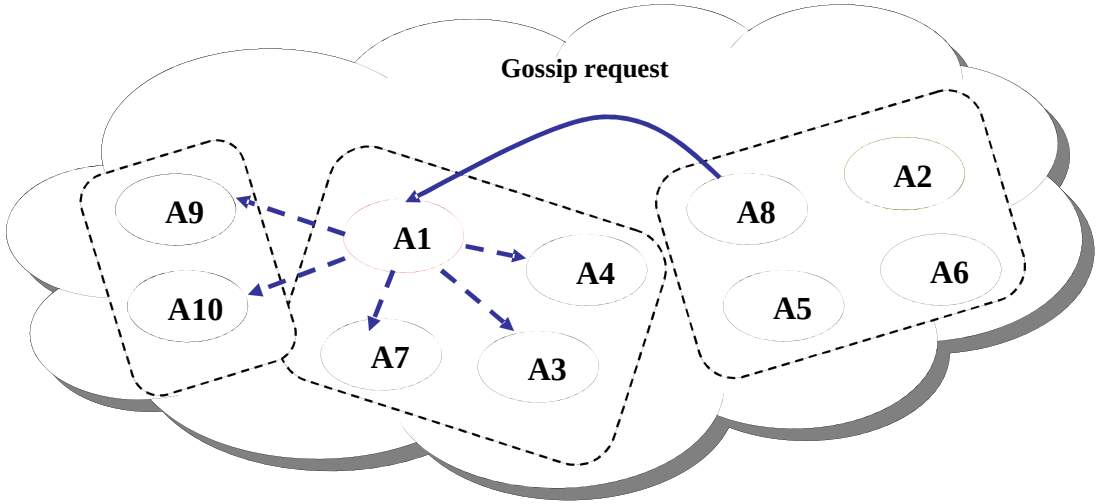


Gossiping (1)


Gossip group
Node 0
Node 1
Node 2
Node 3
Node 4
Node 5
Gossip group
Node 0
Node 1
Node 2
Node 3
Node 4
Node 5
Gossip group
Node 0
Node 1
Node 2
Node 3
Node 4
Node 5
Phase 1
Phase 2
Phase 3
Phase 4
Phase 5


Local *gossip* list
serviceID
suspicionLevel1
time1
+
Received *gossip* list
serviceID
suspicionLevel2
time2
Updated *gossip* list
serviceID
(time1 < time2) ?
min(suspicionLevel 1, suspicionLevel2) :
suspicionLevel1
time

Gossiping (2)



List of monitored objects

Received *gossip* list

Monitored object

+

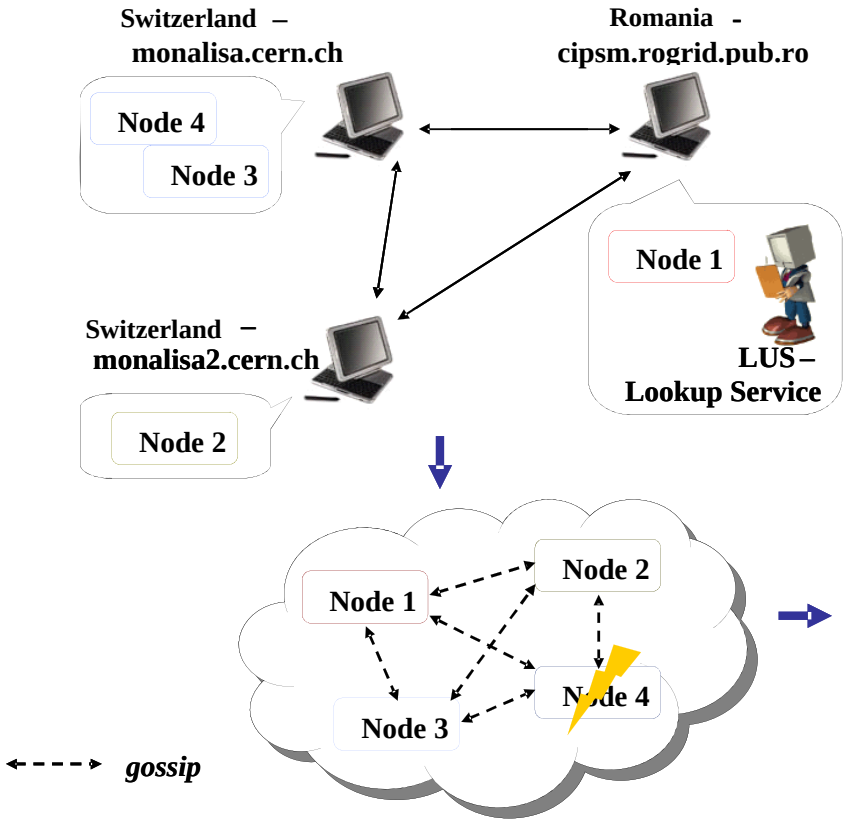
serviceID
suspicionLevel
time

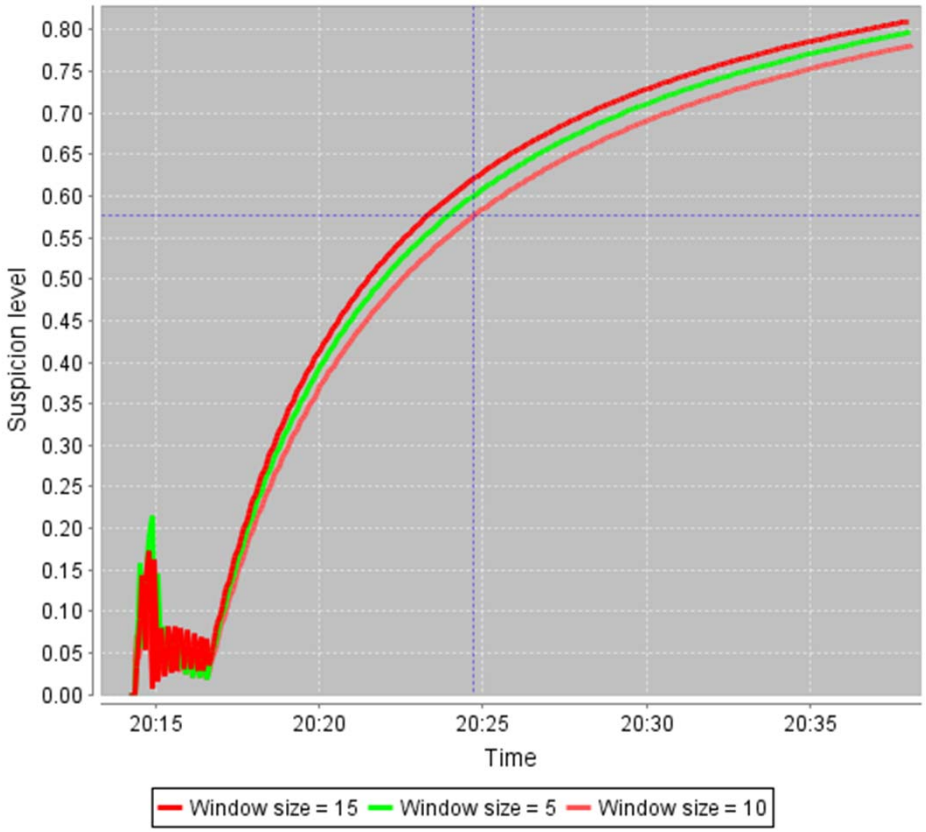
Update of suspicion levels based on received gossip messages

Notification sent to applications

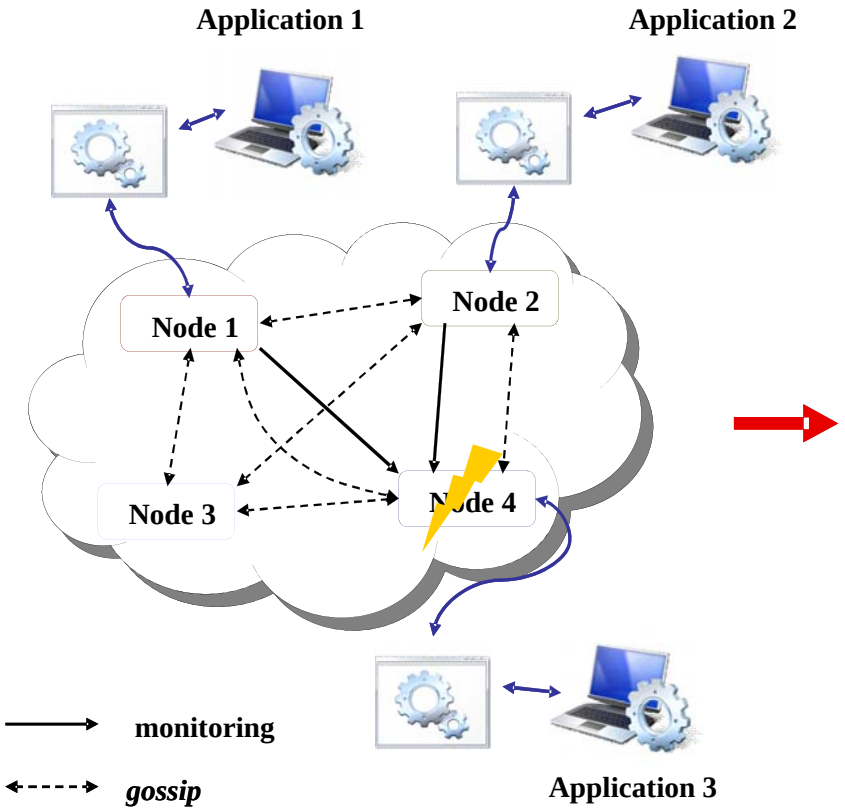
Check suspicion levels

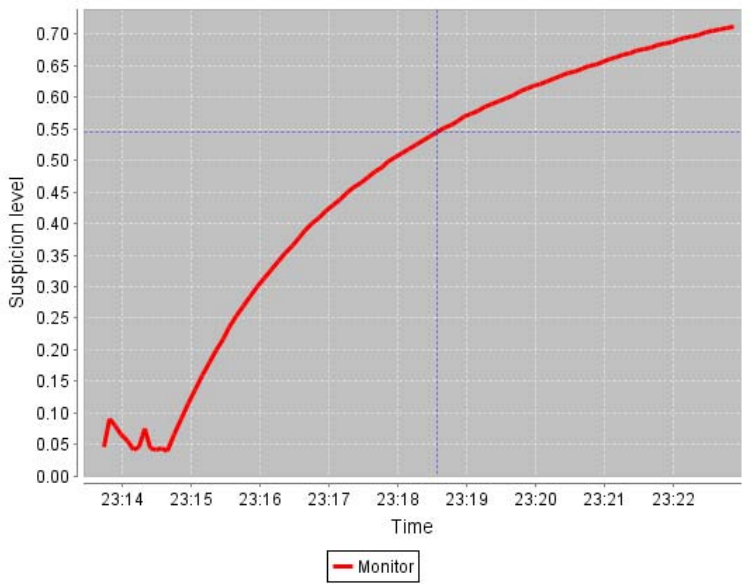
Evaluation results

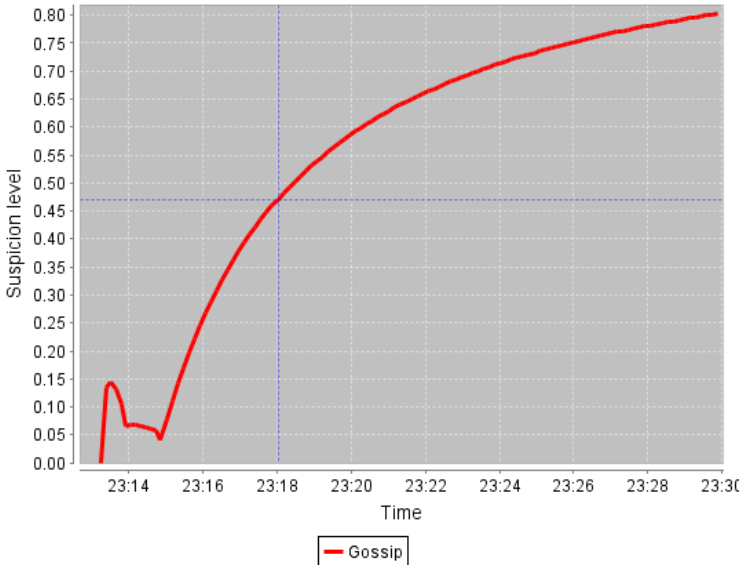




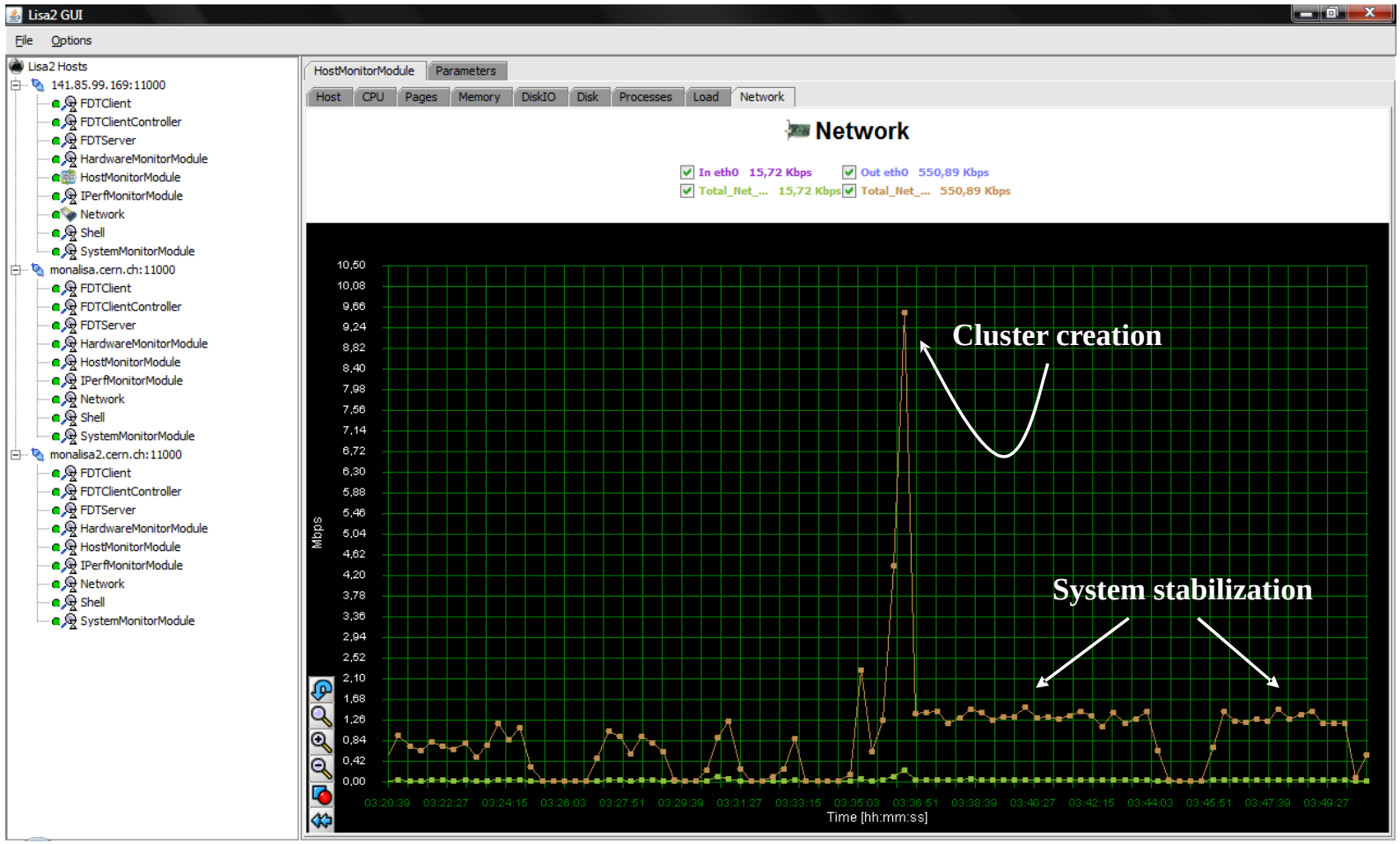
Evaluation results (2)



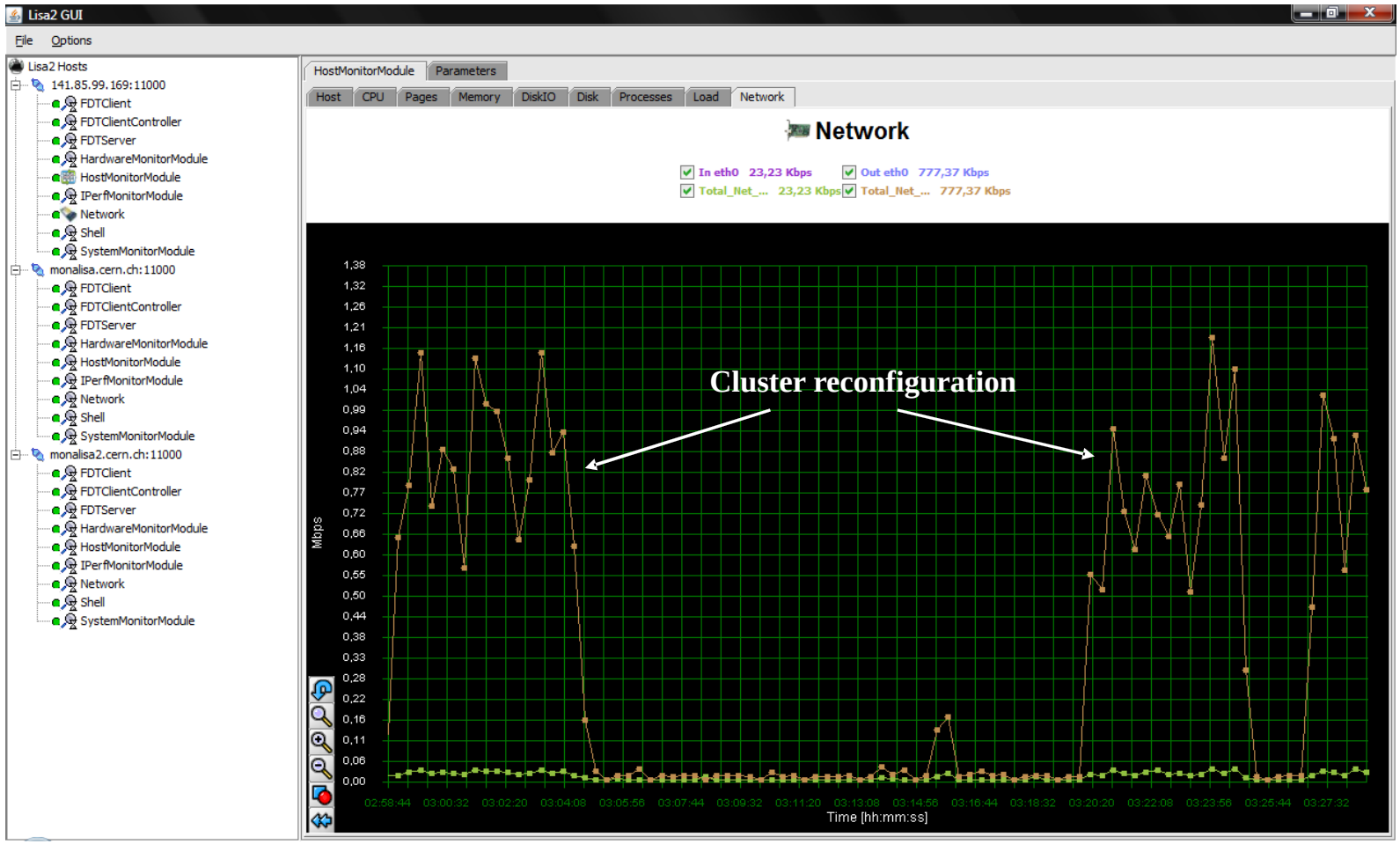




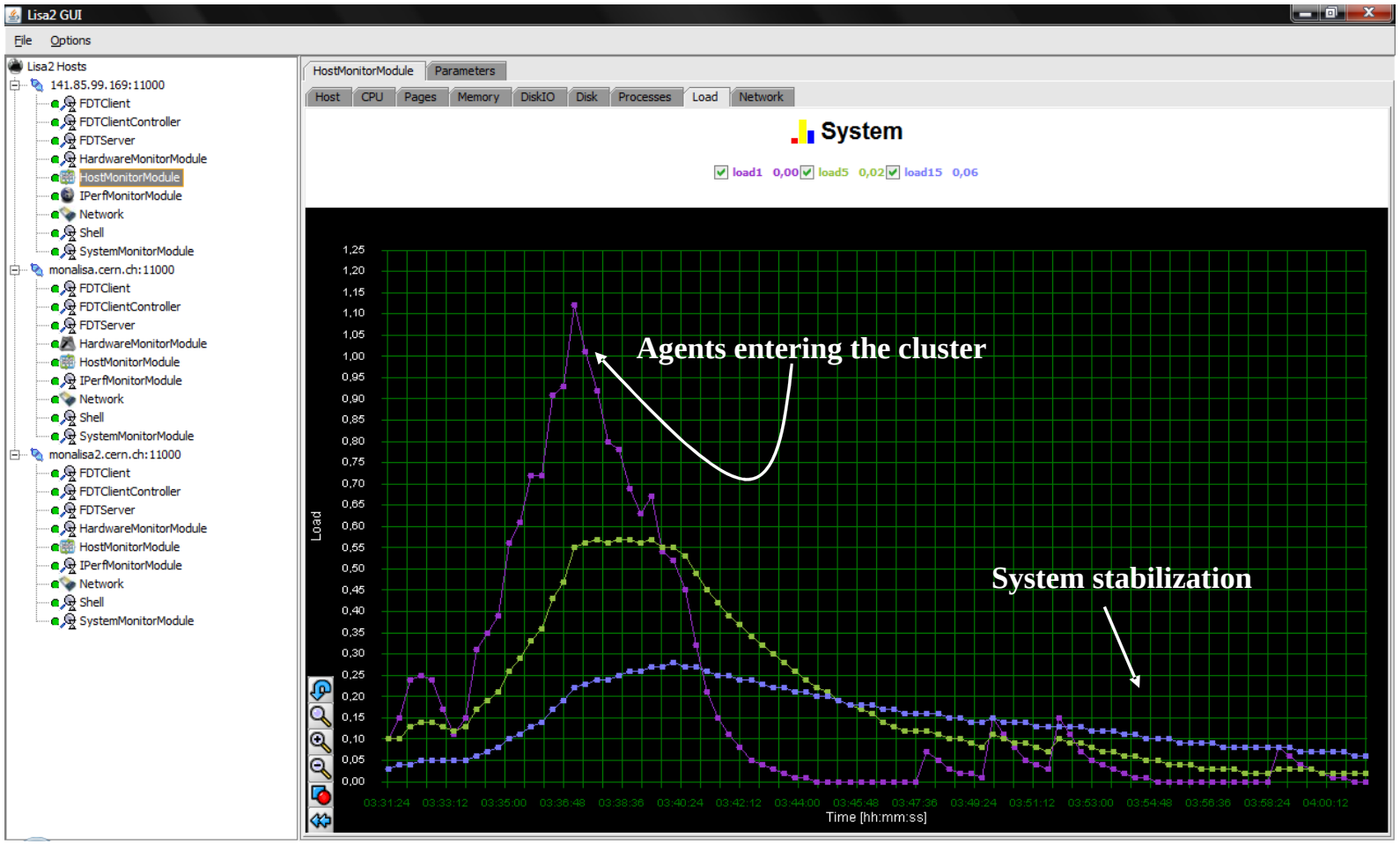
Evaluation results



Evaluation results



Evaluation results



Conclusions

- Robust failure detector that combines fast propagation of information (gossip-based failure detection) and decoupling of monitoring and interpretation (accrual failure detection).
- Advantages:
  - ➔ better estimation of the inter-arrival times of heartbeat messages
  - ➔ increased level of confidence in the suspicions of processes being lost
- Considerations to networking conditions and the QoS detection requirements coming from applications.
- Fast adaptation to reconfigurations and scalability
- Results prove low loads on hosts and decreased network traffic

Questions ?

Thank you!

ciprian.dobre@cs.pub.ro